NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

TECHNIQUES FOR AUTOMATICALLY GENERATING BIOGRAPHICAL SUMMARIES FROM NEWS ARTICLES

by

Matthew W. Esparza

September 2007

Thesis Advisor:	Craig Martell
Second Reader:	Kevin Squire

Approved for public release; distribution is unlimited

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			*Form Approved OMB No. 0704-0188*
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.			
1. AGENCY USE ONLY ***(Leave blank)***	**2. REPORT DATE** September 2007	**3. REPORT TYPE AND DATES COVERED** Master's Thesis	
4. TITLE AND SUBTITLE Techniques for Automatically Generating Biographical Summaries from News Articles		**5. FUNDING NUMBERS**	
6. AUTHOR(S) Matthew W. Esparza			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000		**8. PERFORMING ORGANIZATION REPORT NUMBER**	
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) Department of Defense 9800 Savage Road, Ft. Meade, MD		**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**	
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.			
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited		**12b. DISTRIBUTION CODE**	
13. ABSTRACT (maximum 200 words) The work of manually creating a biographical summary from multiple information sources is both time-intensive and detail-oriented. Automating the task is also non-trivial because of the many NLP areas that must be used to efficiently extract the relevant facts. Yet, no study has been done to determine how powerful a biographical summarization system must be in order to achieve the basic goal of filling slots in a biography template. Equally important, the simplest approaches to discovering and extracting biographical information from text have not been implemented. Further, no standard evaluations have been developed for summarization in general, but an evaluation methodology for this research is described and performed.			
14. SUBJECT TERMS Automatic Biography Summarization, Biography Generation, Dossier Creation, Term Frequency, Multi-Document Summarization, Automatic Summarization			**15. NUMBER OF PAGES** 211
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UU

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

TECHNIQUES FOR AUTOMATICALLY GENERATING BIOGRAPHICAL SUMMARIES FROM NEWS ARTICLES

Matthew W. Esparza
Civilian, Department of Defense Student
B.S., The Master's College, 2004

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
September 2007

Author: Matthew W. Esparza

Approved by: Craig Martell
Thesis Advisor

Kevin Squire
Second Reader

Peter Denning
Chairman, Department of Computer Science

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

The work of manually creating a biographical summary from multiple information sources is both time-intensive and detail-oriented. Automating the task is also non-trivial because of the many NLP areas that must be used to efficiently extract the relevant facts. Yet, no study has been done to determine how powerful a biographical summarization system must be in order to achieve the basic goal of filling slots in a biography template. Equally important, the simplest approaches to discovering and extracting biographical information from text have not been implemented. Further, no standard evaluations have been developed for summarization in general, but an evaluation methodology for this research is described and performed.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

I. INTRODUCTION....1
- **A. BACKGROUND....1**
- **B. STATEMENT OF PROBLEM....2**
- **C. ASSUMPTIONS....3**
- **D. METHODOLOGY....3**
  - **1. Sentence Filtration....3**
  - **2. Redundancy Removal....5**
  - **3. Merge Data....5**
  - **4. Evaluation....5**
- **E. ORGANIZATION OF THESIS....6**

II. BACKGROUND AND RELATED WORK....7
- **A. DEFINITION OF PROBLEM SPACE....7**
- **B. SUMMARIZATION OVERVIEW....10**
- **C. PRIOR WORK....12**

III. RESEARCH DETAILS....19
- **A. GENERAL DESCRIPTION....19**
- **B. INPUT DETAILS....20**
  - **1. PakNews Corpus....20**
  - **2. Fair Isaac....20**
- **C. PROCEDURE....22**
  - **1. System Iterations....25**
    - ***a. Version 1....25***
    - ***b. Version 2....29***
    - ***c. Version 3....30***
    - ***d. Version 4....32***
  - **2. Redundancy Removal....32**
  - **3. Output....33**
- **D. DETAILS OF CODE....33**

IV. EVALUATION COMMENTARY....35
- **A. EVALUATION OPTIONS....35**
- **B. INTERPRETATION OF RESULTS....39**
  - **1. Version 1....39**
  - **2. Version 2....40**
  - **3. Version 3....41**
  - **4. Version 4....42**

V. CONCLUSIONS AND FUTURE WORK....45
- **A. CONCLUSIONS....45**
- **B. FUTURE WORK....46**

LIST OF REFERENCES....49

APPENDIX A – FAIR ISAAC INPUT 53
A. TITLES 53
B. ORGANIZATIONS 91
C. LOCATIONS 115
APPENDIX B 139
A. SUPPORT CLASSES JAVADOC 139
1. Class FileHandler 139
2. Class Person 144
3. Class TupleArray 152
B. INFORMATION EXTRACTION JAVADOC 157
1. Class Ordering 157
2. Class ReferenceCounter 158
3. Class Searcher 164
C. KEYWORD EXTRACTION JAVADOC 166
1. Class KeywordExtractor 166
APPENDIX C 173
INITIAL DISTRIBUTION LIST 195

LIST OF FIGURES

Figure 1. Automatic Biography Generation System. 4
Figure 2. Example Extract and Abstract. 9
Figure 3. Output from entity disambiguation step. 21
Figure 4. Input pre-processing subsystem. 23
Figure 5. Sample output from the system. 33
Figure 6. Chart Displaying Overall Intrinsic Evaluation Results. 38
Figure 7. Precision of version 1 40
Figure 8. Precision of version 2 41
Figure 9. Precision of version 3 42
Figure 10. Precision of version 4 43

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

Table 1. Overview of different versions. 24

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

I would like to thank Craig Martell for his willingness to encourage, advise, and engage in hours of interesting conversation. I would also like to thank my wife, who lovingly supported me through this process. Most of all, I would like to publicly express my gratitude to my Lord Jesus Christ, from whom all blessing and honor come.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. BACKGROUND

While this age may be referred to as the Information Age, it will be left to history to determine how much of the information currently circulated is useful, relevant, and important. The advent of technologies like blogs, MySpace, forums, multimedia sharing sites, and digital outlets for traditional news sources has created an "info-culture" in which every person has the opportunity to be an author with a potential world-wide audience. The challenges to information management that this type of culture presents are apparent. New ways have been and must continually be developed to index and search the vast amount of content available. Additionally, methods of information extraction are needed in the U.S. intelligence community where not just speed, but speed with accuracy is required to return a small number of highly relevant results.

Many researchers wish to see the process of content selection entirely automated based on a user's needs, but the nature of search dictates that humans will continue to be involved. Human involvement dictates that the amount of information presented be reasonably small so that the time to review it and make decisions about how to proceed is small as well. Again, the notion that it is imperative to minimize the time required to review information is a given to entities such as intelligence agencies. Automatic Document Summarization (ADS) is one area of research that has historically focused on techniques for autonomously creating brief abstracts of larger bodies of text. Variations of the technique have surfaced over time to deal with things like creating summaries of multiple document (Multi-Document Summarization or MDS) and generating summaries of non-textual information, such as photographs and other multimedia.

When focused on one genre, online news articles, multiple document summarization techniques can be adapted to find, catalog, and summarize specific types of information. This research attempted to locate and aggregate information about entities and create a biographical summary. To be effective, such a summary would need to be a type of dossier, citing an entity's given name, aliases, title, country, etc. The most

naïve approach to creating such a summary is through the use of arbitrary keyword selectors. For instance, an attempt to find information about an entity's birthplace may include searching the articles for keywords such as "born," or "hometown." This approach quickly snowballs toward a desire to understand the sentence because it cues up questions such as "born to whom?" and "born where?"

Sometimes, summarizing one document may be enough to achieve this goal. Usually, though, it will be necessary to summarize a collection of documents. As stated above, MDS can be used when multiple documents are involved. However, working with more than one document at a time presents new obstacles to summarization such as redundancy removal and conflict resolution. The general problems of summarization like the optimal summary length and what information to include are still present. This research is not interested in optimal size (for reasons that will become clear), but conflict and redundancy are shallowly addressed for the PakNews domain.

Automatic Biographical Summarization, or ABS, is a direct application of MDS, though it may be a slightly easier problem to solve. The problem may be easier because in a robust MDS system, the expected output would be a natural language summary of several source documents of indeterminate length. In an ABS system, however, the expected output is essentially a list of non-prose facts about an entity derived from the source texts. Not only this, in an MDS system the entirety of the document is considered, but in an ABS system the documents are examined on a per-sentence basis. The differences between ABS and MDS will be further explored in Chapter II.

B. STATEMENT OF PROBLEM

Documents containing biographical information are increasingly proliferated as the number of information sources continues to grow. Intelligence analysts must sift through both publicly available and classified resources in order to create profiles of individuals such as foreign heads of state, terrorists, and watch-listed foreign citizen. Assembling this information is a time-consuming and detailed process. No publicly-known system exists to aid analysts with gathering and condensing the information

available. This research attempts to create such a system with the belief that it is faster to assess the accuracy of a computer-generated dossier than it is to generate a full report from scratch.

C. ASSUMPTIONS

Science advances by building upon previous research that has been done in an area. This research is no exception and a necessary prerequisite for the research presented is the tagged output from the Fair-Isaac Entity Disambiguation System. The output and the procedure used to generate it are assumed to be ground-truth. So, in each phase of research, if Fair-Isaac names a particular entity to be an organization, then we will blindly assume this to be the case and develop algorithms for correction if necessary. Also, if Fair-Isaac states that two entities are the same, then they are deemed the same. Details about the Fair-Isaac system can be found in Chapter III.

D. METHODOLOGY

The overall system architecture is summarized in Figure 1 below and provides an overview of how the system interacts with the Fair Isaac system and how the output is produced. The sentences containing the provided references are compiled into a sort of "mini-corpus" for that entity. Further, lists of titles, locations, and organizations were collected from Fair Isaac to prime the information extraction process used later.

1. Sentence Filtration

A secondary goal for this project was to determine the amount of intelligence necessary to extract relevant information from the text. The relevant information which is of primary concern can be categorized as "Name," "Job Title," "Organizational Affiliations," "Lifespan Data," "Quotations," "Family," "Associations," and "Locality Information." To collect this information, three different methods were developed for extracting information from the corpus. Each method looked at the problem from a slightly different perspective and each required a greater amount of complexity. The first method used to extract information was to simply develop several selectors (keywords)

which filtered sentences into the categories seen above. This method was the simplest, though the results derived from it were better than expected.

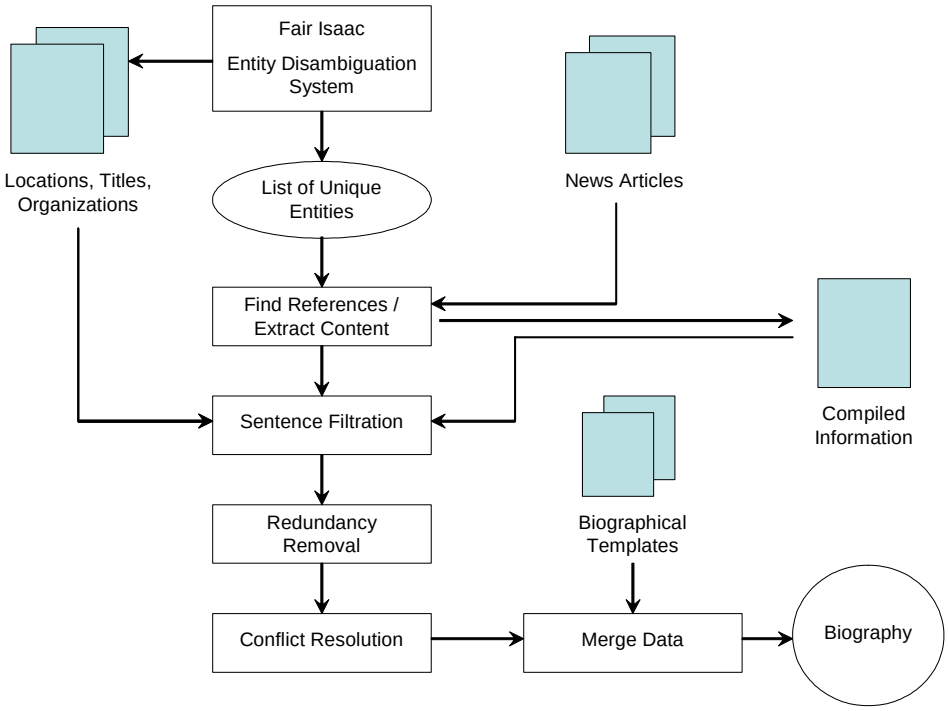

Figure 1. Automatic Biography Generation System.

The second method used slightly expanded on the first method by adding some primitive contextual awareness to the filtering process. For example, when attempting to extract job titles for an entity, it makes sense to look immediately before the entity's name rather than simply collecting all titles that occur in a sentence. This same kind of logic can be used to determine things like associations and family relations.

The third method starts by examining the content of the original marked-up news articles. Then, each time the <PERSON> tag is encountered, a new Person object is created with other details from the sentence or article. These accompanying details (if present) include the timestamp of the article, surrounding entity names, job title of the entity, family information, and lifespan information. The system then merges similar Persons by examining the similarity of the details.

2. Redundancy Removal

A characteristic of news articles is that they often repeat information found in other articles. A key aspect of all four methods is the ability to remove this redundancy from the final information presented in the dossier. The system handles redundancy by performing several recursive comparison and merge operations.

3. Merge Data

A necessary support of this research was the creation of "template" biographies for a type of person. These templates were created because often a corpus of news articles will not contain every detail that can be known about an individual. However, if some characteristics are universally true for a particular type of person, then the known data can be merged with the template data to create a fuller biography. For example, the U.S. President must be at least 35 years old to hold the office, yet that information may never be presented in the source articles. Creating a "U.S. President" template with the age filled in as above allows for introducing real-world knowledge into the process without corrupting the system's decisions. Performing this step can fill in gaps in knowledge and provide an age of *After 1972* for that slot of the biography report.

4. Evaluation

To evaluate the system, the primary concern was the correctness of the information that was gathered and compiled into the reports. Stated differently, the goal of evaluation was to ensure that the facts presented in the different slots of the biography were true facts about the entity in question. A more subjective analysis was also performed to determine the amount of time it took for each report to be compiled by the system as opposed to how long each report would have taken to be compiled by hand. A more traditional strictly statistical analysis was not undertaken for reasons explained in Chapter IV.

E. ORGANIZATION OF THESIS

In this thesis, the background of this work is discussed in Chapter II; a description of the research performed is discussed in Chapter III; Chapter IV explains the results obtained and offers a brief analysis; finally, ideas for further work in this area are discussed in Chapter V.

II. BACKGROUND AND RELATED WORK

A. DEFINITION OF PROBLEM SPACE

A sub-area of Artificial Intelligence (AI) research is called Natural Language Processing (NLP). NLP seeks to find ways for computers to read and write documents in as human a way as possible. Recent departures into the field have included work on applications for searching, indexing, and sorting content of various types. Within NLP, there is also interest dating back to the 1950s in automatically generating summaries of text similar to ones produced by human abstractors. Though interest in summarization waned slightly during the 1970s and 1980s, the advent of the Information Age has created a new flurry of research questions to be solved.

Current media is so summary-laden that the concept seems to be ubiquitous and summary generation is perceived as a trivial research question. Part of the reason for this perception is the ease with which humans both create and decode summaries to glean information. Examples of summaries are all around: television guide synopses, sports statistics, news headlines, stock tickers, advertisements, etc. These summaries are necessitated by the ever-increasing amount of information available. Information volume is also multiplying within the realm of the Department of Defense. A common type of summary that could be of particular interest to the DoD during the current War on Terror is the biography.

A biography is not a summary of text or a set of documents, but of a person's life. Independent of a person's lifespan, the biography is concerned with the events of that lifetime, interactions with others, and affiliations (such as group memberships, racial profile, etc). Knowing these details about an individual can often give a picture of the person's life. For instance, it would be reasonable to make conjectures about when a person lived based upon their contemporaries, their job title, and their parent organization. When used in the context of the intelligence community, documents containing facts like the ones above about a person of interest are called dossiers.

Currently, these dossiers, along with larger intelligence briefs, must be compiled manually, often after the analyst has gone through the long task of gathering intelligence about an individual from various information sources. No known public system exists that can sift through these sources and compile a dossier about a person automatically. Since these sources are most often intelligence reports in natural language, however, the problem becomes one which text summarization can help solve.

When determining how to apply summarization, three things must be considered: input, output, and approach (Barzilay, 03). Even though the focus of summarization work is on text summarization, input can be anything that can be represented as text. This includes audio transcripts, video descriptions, and compiled text from multiple documents. Once the system processes the text, the output can either be a static listing or part of a larger query-based database. The amount of text *displayed* in the output is also variable dependent upon the needs of the application and the content area available.

While input and output are relatively straight forward, the approach (or methodology) to perform the summary is the focus of the bulk of summarization work. There are several ways to categorize summarization methodologies: extract vs. abstract; indicative vs. informative vs. evaluative; generic vs. query based; or single document vs. multi-document. Further, different categories can overlap and add an additional layer of meaning to the method selected.

The terms “extract” and “abstract” deal with the origin of the content used in the summary. If the summary is an extract of a target text, then each word from the summary appears in the source document and has simply been extracted into a compressed form. Unlike an extract, an abstract may be a paraphrase or a completely unique retelling of what the source text relates. Consider this example about the 9/11 Commission Report:

Extract

We present the narrative of this report with a unity of purpose. September 11, 2001 was a day of unprecedented shock and suffering in the history of the United States. The nation was unprepared. How did this happen, and how can we avoid such tragedy again?

Abstract

The 9/11 Commission Report is mainly concerned with the events leading up to September 11, 2001. Various facets of the issue are examined from the teachings of the religion of Islam to the lack of vision by U.S. leaders to the failures of equipment used at Ground Zero. The Commission presents their analysis of the weak points in U.S. intelligence and readiness and proposes solutions.

Figure 2. Example Extract and Abstract.

Summaries can also be categorized according to the content they provide for the reader. There are generally three types of summaries: indicative, informative, and evaluative. Indicative summaries give a clue to the examiner what genre of information the source document contains. Since their job is to simply indicate type of content, indicative summaries can be quite short. Examples of indicative summaries would include the film content information that now accompanies MPAA ratings or the large chapter headings within a document like this thesis. Informative summaries are meant to inform the reader about some fact or related facts contained in the source text. Summaries which are informative are perhaps the most common type of summary and encompass such things as news headlines. Finally, evaluative summaries are meant to be a critical review of the text. A common occurrence of these summaries are book reviews written by consumers on commerce sites such as Amazon.com.

How the summaries are interacted with is also another way to classify what type of summary a system creates. If the summary is a generic one, then it could be intended for a number of applications. Generic summaries could be used to reduce the amount of text necessary for a search engine to index. Also, they could be used to more easily index and classify those texts. Generic summaries are also useful for machine translation, since it is less expensive to translate a smaller block of text than the original document. Query-based summaries are created for the immediate purpose of answering a user query about some information (presumably in a database). The important distinction here is that

query-based summaries do not usually exist before user input is received. Thus, they have the potential to be more dynamic, though fluidity is not required for a query-based system.

The final classification that can be made between summaries is based upon the source text. The source text can either be a single document or multiple documents. Single document summarization present a number of research challenges, such as analyzing discourse structure, selecting salient information, and optimal source compression. Multi-document summarization presents the same challenges, but also adds the need for redundant information checks, cross-document co-referencing of entities, and inter-document conflict resolution.

Once the type of output desired is selected, the input is known, and an approach is settled upon, actual work on the system can begin. The next section will give a detailed overview of what the work of summarization looks like. Then, the third and final section will present a timeline of the work that has been done in the field to date and some commentary on the successes and failures of previous research.

B. SUMMARIZATION OVERVIEW

As stated similarly above, there are three general pieces that every automatic summarization system is composed of: the input, the approach, and one or more output summaries. Again, the input and output are fairly straightforward, but the approach used can vary widely depending upon the intended final use of the system. Almost without exception, the approach employs some type of compression algorithm. In the context of summarization, compression is the term used to describe the process of extracting the most relevant content from the source text. Essentially, the source text is being compressed into a summary void of any information deemed non-essential.

While many compression algorithms exist, the decision to employ one over another is effected by three components: the audience, the function, and the fluency. When performing summarization, the audience can be known (allowing more focused summaries) or unknown. Similarly, the function of the summary can be indicative, informative, or evaluative. The final component of compression is the desired level of

fluency. In some circumstances, one may wish to generate a list of bullet points about a summarized article while other times a more natural language output is desired. After considering these factors, "the compression algorithm will produce one or more output summaries that will be a user-defined percentage of the original source material" (Mani et al., 01).

The summarization process can also be viewed as three different phases: analysis, transformation, and synthesis (Sparck Jones, 97). Analysis can be performed at either the surface level, the entity level, or the discourse level. Intuitively, the surface level of a document is defined as being the actual components; the sentences, images, and headlines, are all major components of the document's surface level. Examining a document at the entity level employs the use of a named-entity recognition (NER) system which can identify persons, places, and organizations. Analyzing a document at the discourse level is the deepest level above a pure semantic parsing of the text. In short, the discourse structure of a document is the flow of meaning in the text which considers things such as anaphora, content from previous sentences, and temporal information and tries to determine how the elements of a document are related.

The goal of the transformation phase is to take any information extracted during the analysis phase and apply algorithms that will fix any word ordering or incorrect grammar situations created. It is important to note that the deeper one analyzes the text, the more complicated the transformation algorithm may become. This growth in complexity is caused by removing smaller pieces from a document that must be plugged into a larger framework rather than extracting larger pieces, like sentences, which can usually be pieced together more easily. A secondary goal of transformation is to remove seemingly unnecessary details from components, such as descriptive phrases, etc., depending on the level of compression desired.

The final stage of the summarization process, the synthesis of the output can be as straight forward as concatenating everything provided by the transformation phase. This is rarely the case, however, because in most summarization situations a close to natural language output is desired. In order to achieve the resemblance to natural language, the

synthesis phase is truly a language generation module that performs content and lexical selection, aggregates phrases into sentences, and creates its own discourse structure (Jurafsky et al., 01).

Each piece of language generation is an area of research all of its own. Content selection deals with examining the content provided by the transformation stage and not only choosing the information that is most relevant from the source text, but also determining what would most aid the summary's coherence. Lexical selection is beneficial to compression because it may be profitable to replace phrases from the source text with single words. For example, the source text may contain a vernacular phrase such as "up the creek" which the generation module could excise and replace with some synonym referring to misfortune. Once content has been selected and all ideas have been expressed in their best form, the text synthesizer must construct sentences from the smaller pieces. This can sometimes be accomplished through the use of heuristics and a sentence template. Once a group of sentences have been formed, they must be chained together to form the summary. Doing this can be very complicated, but can be aided by the use of an underlying discourse structure, which keeps track of the concepts involved in the sentences and uses heuristics about them to cohesively relate them.

C. PRIOR WORK

The foundational application of text summarization was the automatic creation of abstracts for research papers (Luhn, 58). The approach attempted to follow the intuition that the main subject of an article would be a word that appears frequently throughout the article (determiners and prepositions were ignored). To decide which sentences to include in the abstract, the system performed analysis on individual sentences and measured the significance of each one. Significance of a sentence was determined by looking for the presence of significant (i.e., often repeated) words from the overall article. From the smaller set of sentences chosen as significant, a probabilistic algorithm was developed to rank the sentences in order of significance. Sentences that scored above an

arbitrary threshold became part of the abstract. Luhn's work was very influential in directing future researchers to look for statistical techniques when working with summarization.

The next major work on summarization was focused on creating indicative abstracts of scientific articles while also attempting to create an adaptive research methodology (Edmundson, 69). Like Luhn's work, Edmundson attempted to computationally model human abstractors by looking for "significant" sentences. To do this, he expanded the definition of a significant sentence from one that contained high-frequency keywords to sentences that contained *cue words* or *heading words.* He defined *cue words* as words belonging to one of three sub-areas: *bonus words*, *stigma words*, and *null words*. *Bonus words* like "significant" were clues to ideas which were probably essential to the paper's theme. *Stigma words* like "hardly" were defined as words which clued an opposing position to the papers theme. Finally, *null words* did not affect the theme one way or another. The group of *null words* was composed of ordinals, the verb "to be," prepositions, coordinating conjunctions and other less significant parts of speech.

Edmundson also factored in sentence location within a document. For example, a section in his source material may have begun with "In this paper…" This sentence and others similar to it were granted special weights to denote that they probably contained thematic information. Sentences that concluded sections were also given special attention. According to his experimentation, the method which considered locality received higher marks for proper co-selection of information than the other methods tried.

Edmundson's results overall, however, were not promising. The poor performance of his system after seventeen iterations of experimentation led him to conclude that "it is now beyond question that future automatic abstracting methods must take into account syntactic and semantic characteristics…they cannot rely simply upon gross statistical evidence." This result was a hard blow to summarization research in general, because Luhn's work twelve years before had led most researchers to believe that the problem would be quickly and easily solved by more modern algorithms.

After Edmundson, steady work in the area of automatically creating abstracts continued, though interest waned in the second half of the 1970s. The focus continued to be on generating natural language abstracts for scientific (particularly chemistry) articles. This led to stagnation in the field because the vision was completely limited to this small domain which did not provide much material. However, an abundance of material would not have been beneficial at this time, because the methods available to gather data to train and test systems were very costly and time-consuming. Then, once the data was assembled, it had to be analyzed by hand or reviewed by professional abstractors – another timely and costly job.

Summarization research was practically non-existent during the 1970s and 1980s. Instead of attempting to develop actual systems, researchers turned to the more theoretical aspects of the discipline with hopes of making a revolutionary breakthrough. (Paice, 89) posited that there were only seven major approaches for determining sentence significance. They were: frequency-keyword, title-keyword, location, syntactic criteria, cue words, indicator-phrases, and relational criteria. Paice concluded that a frequency-keyword approach is trivial and un-informative. However, he did not completely discredit work based on keyword frequency but declared other word matching approaches such as cue words and indicator phrases as more likely to yield better results.

Out of the rest of the methods, Paice determined that a syntactic criterion (i.e., the distribution of the word throughout the document) was unviable (Earl, 70). This was mainly because the work done by Earl focused on modeling sentences as phrase structure representations and after looking at 3,000 sentences, 99% were unique structures. The progress in the area of Earl's work allows us to now determine that she "overfit" her experiment to her data. Several of the other "new" approaches, such as the frequency-keyword, title-keyword, location, and cue words, were simply expressions of Luhn and Edmundson's previous work compiled in a new form.

Interest in document summarization began to grow again in the 1990s with the advent of the Internet, which affected the field of summarization in two profound ways. First, the Internet provided the opportunity to gather and process much larger collections of data than was previously possible. The foundational research had worked with a

miniscule amount of data (Edmundson worked with only 200 documents) in comparison to what could now be catalogued online. The Internet also provided a fresh need and desire for summarization. As the amount of information on the Internet exploded, new ways were sought to visualize, structure, and organize data.

Kupiec, Pedersen, and Chen (Kupiec et al., 95), developed one of the first modern document summarizers. Their stated goal was to "develop a classification function that estimates the probability a given sentence is included in an extract given a training set of documents with hand-selected document extracts." Their approach was similar to previous work in that it required a pre-compiled corpus of scientific documents and their extracts. Instead of following the past approach of weighting each word and then weighting each individual sentence, however, they used probability and statistics. They derived a Bayesian classification function to assign a score to each sentence based upon distinct features. The score can then be used to determine which sentences to include into a summary. By using a classification function and training on a corpus of documents, they were able to allow the corpus to set the weight of each feature instead of arbitrarily setting the weights in the beginning. The scheme to score features was derived from (Paice, 89). The main evaluation of the system yielded an 83% correctness.[1]

Parallel to the research into creating extracts from text, other research was also concerned with producing abstracts of text. An abstract in the sense of summarization is a short description about an article that contains little or no material from the original article. Another approach was to use a standard template for what a user would prefer an abstract to look like and then to populate the template with material extracted from the text (McKeown et al., 95). Their work became known as the SUMMONS system.

While news of SUMMONS spread, Myaeng and Jang (Myaeng et al., 96) followed Kupiec's approach and created their own probabilistic system. Their approach was slightly modified, however, because they started by manually identifying not only features but components of the text. The sentences were then scored on not just the

[1] For their experimentation, correctness was defined by "the fraction of manual summary sentences that were faithfully reproduced by the summarizer program."

presence or absence of features, but also which textual component it belonged to. Once again, the highest ranked sentences became a part of the final summary.

The next major research in 1999 produced a system known as DimSum (Aone et al., 99). The work was particularly ambitious as evidenced by the introduction of the paper where each approach for generating summaries is chronicled and the discussion is concluded with this statement: "Our work addresses challenges encountered in these previous approaches…" (Aone et al., 99). The team did address several issues. First, the very foundation of the statistical approach was altered by using text statistics and corpus statistics to determine that a word such as "bill" in "reform bill" should be counted independently of "bill" in "Bill Clinton." Second, instead of using Kupiec's scheme of finding entities based upon capital letters, they were able to use SRA's NameTag™ (with accuracy in the mid-90%) to tag their corpus with names, places, etc. Third, the team took advantage of WordNet (Miller et al., 90) to find synonyms to words in the text. This allowed them to bolster the counts of some words by including synonym counts. Even when using some of these advanced tools, however, the final summary was still created by selecting sentences with high scores. Also, scores were still generated by the counts of individual words which make up the sentence.

Still, most of the focus remained on creating generic summaries until others began to research producing biographical summaries (Schiffman et al., 01). Acknowledging that book-length biographies are beyond the capability of computers, the researchers focused on creating a short paragraph containing biographical information from a corpus of 1,300 news documents about the Clinton-Lewinsky affair (termed the Clinton corpus). The approach did not pre-suppose the presence of any particular information in the corpus, but it instead allowed the corpus to dictate what was stated about any particular entity. Also, the research did not take temporal cues into account, which has only recently been addressed (Bethard et al., 07). To generate output, canned text was used to fill in the gaps between extracted texts. While the results and methodology of the paper are generally un-impressive, the most important contribution made to summarization research was the creativity to envision a biographical summarization system.

A second approach to automatically generate biographies used entity recognition coupled with an ontology (Alani et al., 02), or "a set of distinct objects resulting from an analysis of a domain" (Martin et. al, 01). In this case, the researchers sought to automatically generate biographical summaries of famous painters from information found on various reputable Internet sites. The system marks the first attempt to directly interface a biography generation system with the Internet. It was also the first system that sought to produce dynamic summaries for a query-based system in which the user could choose what "view" of the biography they wished to see (where each "view" focused on a different aspect of the author's life.)

While their final output appeared impressive, there are a few subtle points about their research that lessen the luster. First, the researchers chose to follow Schiffman's example and use templates to render the final biography. Second, they made their system heavily domain dependent by developing an ontology to fit the data that their system would be processing. Third, the actual procedure used to extract information from other web sites, which the system uses as a dynamic corpus, is left to the reader's imagination.

These early attempts to automatically generate biographies are examples of systems that dealt with the elements of the source text at an entity level. In information retrieval theory, entities "are things of interest; one might say objects of interest…the objects that a system is designed to store and retrieve" (Smiraglia, 02). Thus, in light of the desire to extract information about persons from news articles, entities are primarily formal names of individuals, locations, and organizations.

Recognizing entities in open text is now done fairly accurately.[2] Yet determining which entity names in the text are part of the set of names used to refer to a particular person in real life is a separate issue known as automatic entity disambiguation, or cross document co-referencing. One system that performs automatic entity disambiguation was developed at Fair Isaac (Blume, 05). In a corpus of Pakistan News Agency articles, the Fair Isaac system was able to achieve greater than 95% accuracy when determining named entities and the algorithm used to merge two entities achieved over 99% accuracy.

[2] Evidence of this can be found in a recent presentation from Microsoft Research: http://www.mathcs.emory.edu/~eugene/talks/cikm2005.ppt.

Merging entities involved resolving many-to-many relationships to determine, for example the difference between references to a cricket player named Yasir Arafat and the deceased leader of the PLO by the same name.

Zhou, Ticrea, and Hovy (Zhou, 05) attempted to address biographical summarization in light of the fact that information is being extracted from multiple documents. They followed the methodology of Aone, choosing to use information retrieval and classification techniques to extract information from their source corpus. To make the final output of their system useful, they also chose to store the biographies and create an interface through which users could query the data.

To train their information retrieval algorithms, they annotated a corpus of 130 biographies about 12 different individuals. Through doing so, they discovered several common components of each biography: lifespan data, popularity, personality, personal, social, education, nationality, scandal, and work. Each sentence in the corpus utilized was classified with one of the above labels and also labeled as to whether it belonged in the final biography or not. The sentence's presence in the final output was determined by its score, which was again determined by textual and corpus statistics.

The most recent research into biographical summaries focused not on creating a full narrative about a person's life, but on answering biographical questions about a person (Feng et al., 06). The output they expected from their system would accurately answer questions such as "When was Albert Einstein born?" based upon information extracted from web pages. To aid efficiency, they proposed to use data mining to gather information ahead of user queries and cache the answer for later use.

The current state of the art in biographical summary is not clearly defined. It depends entirely upon the focus and domain of the research. Natural language biographies have been generated, but only using templates and canned text. Query based systems have been developed, but the answers returned are generally tidbits of a biography and not the entire generic biography. Further, evaluation of summarization systems in general, including biographical summarization systems, remains an open research question.

III. RESEARCH DETAILS

A. GENERAL DESCRIPTION

Automatic summarization continues to evolve as a complex and multi-faceted problem that is deservingly receiving much academic attention. Yet, as demonstrated in the previous chapter, very few researchers are approaching the problem of automatic biography summarization (ABS). Perhaps more troublesome, the current line of biography summarization is perceived as a completely parallel line of research to automatic document summarization. Thus, researchers automatically apply the latest document summarization research to ABS work to ensure the construction of robust systems. These "robust" systems, however, are generally limited to one domain and one function and have been given far more complicated intelligence than what is actually needed to accomplish the task.

While many techniques used in document summarization should undoubtedly be applied directly to ABS work, the fundamental question about how powerful a system would need to be to perform ABS has been left unanswered. Are full discourse analyses, named entity recognition, and robust semantic classifiers necessary or can the job be performed using mostly keyword selectors and some knowledge of the target documents?

Since that basic question had not been answered in the literature surveyed, the hypothesis which guided experimentation was that keyword filters and word location would be sufficient to achieve a system with a very high degree of information integrity. In order to test this hypothesis, four versions of the system were developed to perform the keyword filtering in different ways. The different versions represent the "approach" piece of the summarization process, composed of the input, approach, and output.

B. INPUT DETAILS

Each of the four versions of the system used the following input sources:

1. PakNews Corpus

Put simply, a corpus is a collection of texts (Saggion, 04). When Fair Isaac sought to build their automatic disambiguation system (Blume, 05), they needed to collect a group of documents that would be well suited for their purpose. They decided to use news articles, which is logical given that news articles are generally full of references to different entities. In particular they collected articles over a 44 month period from the Pakistan News Service. The primary reason why this particular service was chosen was because the PakNews Service is written mostly by amateurs who may have differing levels of expertise transliterating Arabic names to English. This allows for a single name to be spelled (and misspelled) multiple different ways – an ideal challenge for a disambiguation system. The articles are also conveniently accessed via the Internet at http://www.paknews.com[3] and each article has an accompanying timestamp.

2. Fair Isaac

The system provided by Fair Isaac is primarily written in Perl with a user interface coded in TCL/Tk. The overall goal of the system is to disambiguate references to people with the same name. Within the field of natural language processing, this is referred to as cross-document co-referencing. The system creates feature-based vectors and compares them to each other to determine whether one entity is the same as another. For instance, in the PakNews corpus, there are two entities referred to as Yasser Arafat. One Arafat was the head of the Palestinian Liberation Organization while the other is a cricket player. Originally developed for use in determining a person's credit score, the system is able to disambiguate the references with very high overall accuracy (greater than 95%).

[3] At the time of this writing, this site is no longer active.

While the full explanation of how the disambiguation is performed can be found in (Blume, 05), it may be profitable to describe the process in general. The original PakNews corpus has been tagged with XML to identify document elements such as headlines, document boundaries, and document timestamps and mentions of named entities. Each entity mention (person, organization, and location) is given an identification number and placed into a XML master list of corpus entities. This list is processed by the system and produces a new XML list of disambiguated entities, complete with a list of all identification numbers that are the same entity. These identification numbers are pointers back into the original articles where the entity of interest is mentioned. An example of the output of this processing step can be seen in Figure 3.

```
<PERSON ID="p_osama_laden_001_p" LNG="Osama bin Laden">741
1568 1907 2268 3720 4294 4427 4643 4706 4750 4871 5508 6094
6336 6920 6931 7041 7245 7417 8654 8973 9010 9273 9430 9446
9584 9632 9687 10209 10238 10303 10307 10903 11338 11418
11446 13800 14582 15032 15385 15484 15550 15666 15709 16018
17922 18143 18473 18524 21125 21225 21376 21378 21381 21382
21422 21434 21518 21538 21540 21554 21557. . .
```

Figure 3. Output from entity disambiguation step.

These mentions are traced to their original positions within the documents and the enclosing sentence is extracted as a unit. Each entity mention appears in the original news articles in the following way: "<PERSON ID="**741**" STD="**p_osama_laden_p**">**Laden**</PERSON> **allies suspected behind Buddha destruction.**" Conveniently, the original news document data files place one sentence per line which is terminated with a newline character. Thus, there is no need for complex sentence tokenization past splitting the string based on the appearance of newline characters.

As stated above, the PakNews corpus also contains occurrences of organizations and locations tagged in a similar manner to persons. Using this tagged corpus along with the entity mention IDs, the preliminary conclusion was that finding information about

those entities would be relatively easy. Thus, the information from the disambiguation system was used as input to the biography generation system. The lists representing the input received are listed in Appendix A.

C. PROCEDURE

Each version of the system was written in Java and developed using the Eclipse IDE. Though most data structures were custom-built, some relied heavily upon the built-in Java regular expression, input/output, and hash classes.

When working with the data from the Fair Isaac system, several pre-processing steps were necessary before the work of information extraction could begin. To pre-process the input, several data files were created to allow for easier handling. Since the corpus being used was a static collection of news articles and had already been processed by an named entity recognition (NER) system, several categories of data were considered "closed" (i.e., the world existed of only what the NER identified). These closed data categories included titles, organizations, and locations.

The members of these different categories were each placed in separate files and loaded into the biography summarization system as arrays of patterns. Meanwhile, several classes were developed to handle the data and perform the work of summarization. A diagram of the sub-system developed for pre-processing the data is provided in Figure 4.

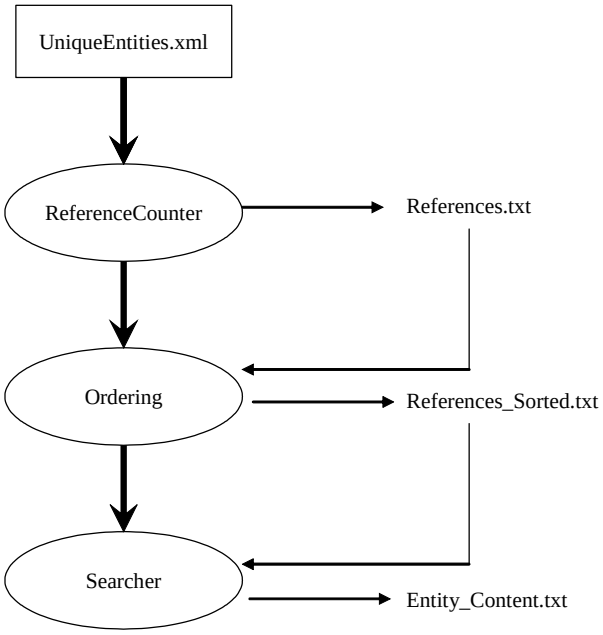

Figure 4. Input pre-processing subsystem.

The *ReferenceCounter* class processes the Fair Isaac system's output, UniqueEntities.xml. Specifically, it processes the lists of mentions for each entity. If an entity does not contain an arbitrary number of mentions, then it is discarded. The class creates a new file to contain the new list of entities and adds some information, such as the entity's name (according to Fair Isaac), markers denoting the beginning and end of mention references, and a count of how many entities were returned.

As stated above, the output of the disambiguation process is a long list of entities and their references. The disambiguated list originally contains 55, 695 persons.[4] Of that number, 46,498 persons are only mentioned once while only 125 are mentioned more than 100 times. For the purposes of this research, only content about persons mentioned in the articles 100 times or more was compiled. The reasoning behind this stemmed from the desire to avoid a sparse data problem and will become more apparent when the output is discussed later in the chapter.

While *ReferenceCounter* will print the references for each entity in numeric order, the listing of entities is still sorted alphabetically. The *Ordering* class puts the entities in

[4] From this point on, the reference to persons will technically refer to a person entity.

numeric order to make searching the corpus easy. *Searcher* does just as its name implies by sweeping through the corpus to find PERSON tags. Since all PERSON tags in PakNews have a corresponding identification number, and since these numbers are in numeric order throughout the corpus, the list of entity reference mentions can be processed at the same time as the corpus. If the PERSON tag contains an identification number that is currently being looked for, then it is added to the possessing entity's block of information. The final output from this step is passed to each version of the system.

While each version of the system is concerned with matching keywords in the text pertaining to each entity, they each go about it in slightly different ways. Each version, though, does make final decisions about what to declare as an entity's job title, organization, and location based upon the *argmax* over the counts of the elements (explained below). Table 1 presents an overview of each system.

Version Number	**Approach**
1	Primitive keyword matching. No cue words or word locations were taken into account. No redundancy removal.
2	Keyword matching with attention to word location and cue words. Wholly discarded sentences that were thought to be quotations. No redundancy removal.
3	Keyword matching with attention to word location and cue words. Used basic information (job title, organization, location) found in quotation. Decided city based upon country.
4	Keyword matching with attention to word location and cue words. Modified name filter to increase precision. Excluded false familial phrases such as "father of the nation" from the familial filter.

Table 1. Overview of different versions.

1. System Iterations

a. Version 1

During processing, the persons' references and their content were held in a large array of Person objects. The overall system algorithm dictated an approach that iterated through the array of Person objects and ran each of the filters below on each of the sentences in the current person's content block from PakNews. Once the filtering of information was complete for all of the sentences, decisions about an entity's name, title, organization, and location were completed and stored before moving to the next entity. The sentence filters and the decision processes are now discussed in more detail:

Entity Name Filter. A simple name composed of the entity's first and last name was obtained from the Fair-Isaac Entity Disambiguation System. This name is the basis for a pattern of what the system will look for in the sentences which mention the entity, Se, and is also expanded upon to look for any additional middle names. For each mention, m, of an entity, e, found in Se, we can compute a count of the occurrences of m:

$$count(m) = \sum_{m \in S_e} 1$$

Each mention in S_e that does not correspond to a previously discovered mention is added to a list, L_n, which represents the possible names that could be assigned to an entity. For instance, U.S. President George Bush could rightfully be mentioned in text as G.W. Bush, G. Bush, George W. Bush, etc. Formally, the list is defined as the following:

$$L_n = \{m \mid ((m = e) \wedge (m \in S_e) \wedge (m_i \neq m_j))\}$$

To decide which version or format of a name is correct for a particular entity, we *argmax* over the counts of the mentions in L_n.

$$Name_e = \arg\max_{m \in L_n} count(m)$$

The decision reached ($Name_e$) is the final name published in the biographical summary. Note that this approach does not take into account misspellings in the name or variations of the same name. Instead, it treats each first and last name pairing as unique.

Entity Title Filter. Before filtering sentences, we collect all of the possible titles from the corpus. Again, the Fair-Isaac system allows us to bootstrap our implementation by making determinations for us about what is a title and what is not. After creating a list of titles, we look for those instances when a title, *t*, is used in the entity's content S_e. A running tally (computed below) of how many times each title is found is stored for later analysis.

$$count(t) = \sum_{t \in S_e} 1$$

Each title and count pairing is stored in a list, L_t, defined as:

$$L_t = \{t \mid ((t = e_t) \wedge (t \in S_e) \wedge (t_i \neq t_j))\}$$

To arrive at a decision of what title to assign to a person, we again *argmax* over the counts of the titles collected for that person.

$$Title_e = \arg\max_{t \in L_t} count(t)$$

The final title decision is then published in the final biographical summary.

An interesting note about titles in the PakNews corpus is that Arabic titles sometimes imply a deeper meaning than just a person's position. For example, the title Maulvi is used to denote a Sunni Muslim religious leader. Thus, from a title we may be able to initially deduce a person's religious orientation and in the example above, the specific sect of religion adhered to. In a non-Arabic context, a title such as "Commander-in-Chief" may be a reference to the President of the United States and provide information about a person's range of influence and their nationality.

Entity Organization Filter. Similar to the titles from the PakNews corpus, all possible organizations are identified and compiled into a single list. Each organization name is then searched for in the set of sentences pertaining to the entity, defined as S_e.

When an organization, *o*, is located in the content, it is added to a list, L_o, of possible organizations to which the current person may belong.

$$L_o = \{o \mid ((o = e_o) \wedge (o \in S_e) \wedge (o_i \neq o_j))\}$$

If an organization is encountered a second time, a count for that particular organization is incremented. Formally,

$$count(o) = \sum_{o \in S_e} 1$$

To then determine an entity's organization affiliation, we *argmax* over the number of counts for each organization and assign the one with the highest count to the entity.

$$Organization_e = \arg\max_{o \in L_o} count(o)$$

This decision is published in the final biography as the organization choice for a particular entity.

Entity Location Filter. Locations in the PakNews corpus are tagged as such. Again, a listing of these locations can be compiled and searched for in each entity block, S_e. In the case of locations, however, cities and countries show up as a pair of strings separated by a comma (e.g., Islamabad, Pakistan). So, instead of deciding the location as a unit including both city and country, the location string is split in two. As before, the cities and countries found in the relevant content are associated with a person as possible locations. A count is maintained for each location (i.e., city or country) found. For cities,

$$count(lci) = \sum_{lci \in S_e} 1$$

For countries,

$$count(lco) = \sum_{lco \in S_e} 1$$

Each possible city location is added to a list of cities, L_{lci}, while each possible country location is added to a list, L_{lco}:

$$L_{lci} = \{lci \mid ((lci = e_{lci}) \wedge (lci \in S_e) \wedge (lci_i \neq lci_j))\}$$

$$L_{lco} = \{lco \mid ((lco = e_{lco}) \wedge (lco \in S_e) \wedge (lco_i \neq lco_j))\}$$

Then, we *argmax* over the counts of the cities and the countries and arrive at an independent decision for each.

$$Location_{e_{ci}} = \arg\max_{lci \in L_{ci}} count(lci)$$

$$Location_{e_{co}} = \arg\max_{lco \in L_{co}} count(lco)$$

This allows the system to come to a decision about the city independently of its decision about the country.

Entity Quotation Filter. The most basic approach to capturing quotations was based upon matching the double quote character (“). The implication of doing this was that words placed inside quotations for a purpose other than quotation were captured and direct quotes that were not in quotations were not captured. To combat this, additional expressions were added to look for clue words such as “said” or “says” that indicate the presence of at least an indirect quote. Since quotes in news articles are said by individuals with their own agenda, the information they contain cannot necessarily be treated as fact. Eliminating facts contained in quotations is slightly troublesome, since the conditions arise that the first-hand sentiment of someone who witnessed a terrorist attack performed by an entity is ignored because it appears as a quotation. This elimination risk is minimized somewhat by the high volume of redundancy in PakNews. They are still valuable for determining an entity’s frame of mind, however, and can be useful to a dossier. For instance, the only evidence of an entity’s intention to commit a terrorist attack may be contained in a quotation. This possibility makes their inclusion in the final biographical summary essential.

Entity Familial Relationship Filter. This filter seeks to uncover the names of spouses, children, descendants, and ancestors. To find these relationships, keywords and phrases such as "father," "father of," "mother," and "in-law," are searched for in the source text. If such keywords are discovered, the sentence in which they were embedded is extracted and made apart of the entity's final biographical summary.

Entity Professional Relationship Filter. Meetings, conferences and debates between other persons and organizations characterize professional relations. A less popular professional relation exists between persons and organizations when the former lectures or gives a speech to the latter. This is a weaker relationship because an organization is made up of many members, all of whom may or may not hold certain things in common with the person. To find information about professional relationships, keywords like "meeting," "met with," and "addressed" are searched for in the source text. As when searching for familial relationships, if keywords are discovered, then the entire sentence is extracted and added to the entity's biographical summary.

Entity Lifespan Filter. The lifespan filter deals with the birth and death dates of an entity. It can also collect the dates of the articles, providing a fixed reference point in time to base judgments about relative temporal information in the article (e.g., last Thursday, yesterday, etc). Though the parsing of temporal information does not take place in any version of the system, strings matching a date format (e.g., 5-12-1980; May 12, 1980; etc.), and keywords referring to life and death such as "born," "died," and "birth" were searched for in the relevant source text. Again, sentences containing the keywords are extracted and added to the final biographical summary of the individual.

b. Version 2

The initial results received from version 1 precipitated the modification of several filters to adjust the amount of information that was being collected by the system. The decision processes remained the same, but the content being fed into those processes was changed by modifying the filters. The changes to the filters affected is described below:

Entity Title Filter. Instead of simply looking for all co-locations of titles with the entity's name, the filter only captures the title if it appears immediately before the first or last name of the entity. This allows the filter to have some concept of context and also allows for consideration of word ordering. In broad terms, word ordering refers to the syntactic structure of a sentence (i.e., the order of the words). As titles from the global list were found in the source text, they were added to a list of possible titles. As in version 1, we then *argmax* over the counts of the titles and choose the resulting string as the most likely title of the entity. This title appears in the final biographical summary of the entity.

Entity Quotation Filter. While the original approach insisted on collecting quotations and assigning them to the source, this proved extremely difficult to evaluate and made the system fairly inefficient. Further, it could be argued that quotations reflect opinion and therefore cannot be trusted as sources of truth of any kind. For these reasons, the quotation filter became a litmus test for sentences. If they contained a quotation, then that was noted and the sentence was skipped.

c. Version 3

On the system's third iteration, some efficiency issues were addressed. Class files were written for Organizations and Locations which became a wrapper for the array of patterns and strings that had been used previously. Instead of using an array of Persons to hold information during execution, however, one reusable Person object was created and the content was printed to the file after the current person was processed. There were also changes to different filters and the information collected was processed by a function that removed redundancy (described in the next section).

Entity Name Filter. Some names provided by the entity disambiguation system were composed of only one name as opposed to a first and last name. Thus, the words before and after the given name were examined to see if any dominant name emerged. The collection and decision process were unchanged from version 2 of the system.

Entity Organization Filter. In the list of organizations recovered from the entity disambiguation system's output, organizations were paired with their corresponding acronym. Previously, just the organization name was searched for, but this was changed to look for both the presence of an organization's name and/or the presence of the organization's acronym. When deciding the organization, the instances of the organizations acronym were attributed to the score of the organizations full name.

Entity Quotation Filter. In the previous two versions, if a sentence was found to be either a formal or an informal quotation, it was ignored. The reasoning for this was based on not wanting to pollute any decisions made with opinion. The impact on ignoring these quotations was examined, however, and it was discovered that including them in the data to influence the system's decision of the person's name, job title, organization, and location was beneficial.

Entity Location Filter. The entity location filter had been separated to decided country and city of an entity independently until this version. As locations were encountered in the text, the tally is still calculated as before:

For cities,

$$count(lci) = \sum_{lci \in S_e} 1$$

For countries,

$$count(lco) = \sum_{lco \in S_e} 1$$

These locations were then compiled into two lists, L_{lci} and L_{lco}, defined above. In version 3, though, the functionality was merged to base the decision of the most likely city upon the decision made about the most likely country. Once the most likely country was determined, the most likely city within that country was chosen to complete the person's most likely location. Formally,

$$Location_{e_{co}} = \arg\max_{lco \in L_{co}} count(lco)$$

$$Location_{e_{ci}} = \arg\max_{lci \in L_{co}, C} count(lci)$$

where $Location_{e_{co}}$ is the country slot of the entity biography, $Location_{e_{ci}}$ is the city slot of the entity biography, *lco* is each unique country encountered, *lci* is each unique city encountered, and *C* is defined as the set of all possible cities that are located in a particular country.

d. Version 4

Some filters experienced minor changes to improve performance.

Entity Name Filter. The name filter was adjusted to be more precise by accounting for whitespace between the words. The name filter had been looser before and was set to match anything between the first and last name. This allowed for "Abdullah" to be matched for "Abdul." Also, instead of looking at the word before and after one word names, a simple space before and after to delineate the name was added instead.

Entity Familial Relationship Filter. This filter was improved to exclude common phrases found within PakNews such as "father of a nation" from the indicative list of familial relationships.

2. Redundancy Removal

It is assumed that when dealing with news articles that material will overlap from one document to the next. This redundancy subsequently appears in the information collected by the different sentence filters. During experimentation, three types of redundancy were encountered and can be classified as *duplication*, *encapsulation*, and *affirmation.* Duplication is the most basic form of redundancy and the easiest to spot in which the complete sentence appears twice. When duplication was encountered, the second occurrence was removed from the final biography. Encapsulation deals with part of a sentence being inside of another sentence. To manage this case, the smaller sentence

was removed and the longer sentence was preserved. Affirmation is the trickiest type of redundancy and was not addressed by this research. When one sentence affirms another, the information related in the sentence is the same, but the words used may be completely disjoint. This form of redundancy requires a more robust semantic framework to determine similarity and could be an independent research topic.

3. Output

The original vision for the output was a simple text display of facts relevant to different categories of information about the entity. The final version of the output was formatted in XML (for future query based work) and, for some categories, the entire relevant sentence was displayed. A sample of the type of output produced can be seen in Figure 5.

```
<PERSON ID="12">
        <NAME>
                <DECISION>Aftab Ahmed Khan Sherpao</DECISION>
        </NAME>
        <TITLE>
                <DECISION>Mr</DECISION>
        </TITLE>
        <LOCATION>
                <DECISION>Islamabad,Pakistan</DECISION>
        </LOCATION>
        <ORGANIZATION>
                <DECISION>Accountability Court</DECISION>
        </ORGANIZATION>
        <FAMILY>
                <FAM ID="1">Moreover, Sikandar Sherpao, son of
                Aftab Ahmed Khan Sherpao will start his
                parliamentary career from the NWFP assembly
                his time</FAM> ...
        </FAMILY>
        <ASSOCIATES>
                <REL ID="1">Shujaat informed that he would
                meet PML-Q allies including National Alliance,
                Aftab Sherpao and MQM</REL> ...
        </ASSOCIATES>
</PERSON>
```

Figure 5. Sample output from the system.

D. DETAILS OF CODE

A partial listing of core classes is presented in Appendix B. The source code for the fourth and final version of the system can be found in Appendix C.

THIS PAGE INTENTIONALLY LEFT BLANK

IV. EVALUATION COMMENTARY

A. EVALUATION OPTIONS

Evaluation of automatic summarization applications continues to be an unsolved problem (Jing et al., 98). The reason for this is that summarization goals vary so widely that it is almost impossible to determine a universal evaluation for all systems. In general, however, summarization evaluation methods divide into two categories: intrinsic and extrinsic.

An intrinsic evaluation is focused on evaluating the content of a summary. This includes its textual coherence and how useful the information in the summary is to the given task. (Mani, 01) outlines several intrinsic evaluation methods:

a. Summary Coherence – Since most summarization system output is in natural language, either because the information is extracted or because some language generation module has been applied, readability of the end result is a necessary indicator of a worthwhile summary. Information can be extracted in such a way that dangling anaphors exist or the discourse structure of the original text is broken. The coherence of a summary is usually subjectively evaluated by humans who give summaries a grade on a scale determined by the researchers.

b. Summary Informativeness – The summary of a source text can be completely coherent, yet lack substance of any sort. Thus, the information extracted must be analyzed based upon what role the summary is expected to fulfill. Informativeness, like coherence, can be judged subjectively, but it can also be scored according to the standard definitions of precision and recall. Recall in this case measures how many human-created reference summaries contain the same information as the machine-created summary. If the extraction algorithms are sound, a remaining obstacle to informativeness may be the amount of compression of the source

document was required. There is no acceptable way to compare the informativeness of variably compressed summaries.

c. Comparison Against Input – This form of evaluation involves the informativeness of the summary, but can also encompass more than that. The options for performing this type of evaluation are using semantic methods or surface methods. Using semantic methods requires that each sentence in a text be hand-tagged with a meaning. The summary can then be judged by how many topics from the original source it covers. A surface method can be performed by identifying key passages in the source text and then looking for the presence of that content in the summary.

So, while intrinsic evaluation focuses on the actual summary, extrinsic evaluations are oriented around the ability of the summary to make a job easier for humans to perform (relevance assessment (Mani, 01); reading comprehension (Morris, 92); etc). Usually, the impact of a summary system on a job is a reduction in the time necessary to reach the conclusion one would have reached by reading the entire source document. In the TIPSTER SUMMAC evaluation (Mani, 01), which created indicative summaries for articles that were then sifted by government intelligence analysts, the "relevance assessment time [was reduced] by 40% …to 50%, with no statistically significant degradation in accuracy."

The other primary use of extrinsic evaluations, reading comprehension, allows humans to read full length documents or summaries and then answer multiple choice reading comprehension questions. If users reading summaries are able to score as equally high as users reading the full document, then it can be concluded that the summary is highly informative. Similar experiments (Hovy et al., 98) have been conducted in which users must create the original source document based upon the summary.

Broader objective and subjective methods exist to judge summaries as well. Objective methods for evaluating summarization output generally compare human summaries to ones generated autonomously. The most common and oldest method is to

have human abstraction experts create summaries for the content in question and then compare the system's summary based upon similarity of words, content selection, or other characteristic. This group of methods can be time-consuming and costly, however, especially if there is a large volume of content.

More standard statistical evaluations such as calculating the precision, recall, and F-score also fall into this group of methods, though it is not always clear how to define them in this domain. Precision is generally defined as the number of correct items given by a system divided by the total number of items given by the system.[5] In this definition, "items" is left somewhat ambiguous and the term must be quantified for an individual system before an evaluation can be initiated. The general formula for precision (P) is:

$$P = \frac{\text{True Positive}}{(\text{True Positive} + \text{False Positive})}$$

Related to precision, recall is generally defined as the number of correct items provided by a system divided by the total number of correct items in the original text.[6] The general formula for recall (R) is:

$$R = \frac{\text{True Positive}}{(\text{True Positive} + \text{False Negative})}$$

Finally, the F-score is a measurement that balances precision and recall by taking the harmonic mean[7] of the two. The general formula for the F-score (F) is:

$$F = \frac{(2 * P * R)}{(P + R)}$$

Subjective methods focus on assessing a summary's informativeness and coherence. If this is to be done scientifically, it requires multiple reviewers who have established methods for rating the information contained in the summary and how well the information flows (applicable only when creating a summary in natural language).

[5] Jurafsy and Martin. Speech and Natural Language Processing. Prentice Hall 2003, p. 578.

[6] Ibid.

[7] A discussion of the harmonic mean is found here: http://en.wikipedia.org/wiki/Harmonic_mean.

The problem with this set of methods, however, is that reviewers must be familiar with all of the content contained in the original article in order to know if all relevant information is contained in the summary.

Remembering that the biography summarization is attempting to model an automatic dossier generator, there are only a few options when performing an evaluation. An intrinsic evaluation based upon sentence precision can be performed, while a more limited evaluation of sentence recall can also be performed to ensure that precision is not being extended at the expense of recall. Further, an extrinsic evaluation can be performed based upon how much estimated time is saved in the process of automatically generating the biographies.

For the time constraints of this research, it was sufficient to examine 18 biographies[8] from the output of processing the 125 persons that were mentioned more than 100 times in PakNews. The output produced by each version of the system was examined and compared against facts that could be verified from the corpus. The comparative results for each version of the system can be seen in Figure 6.

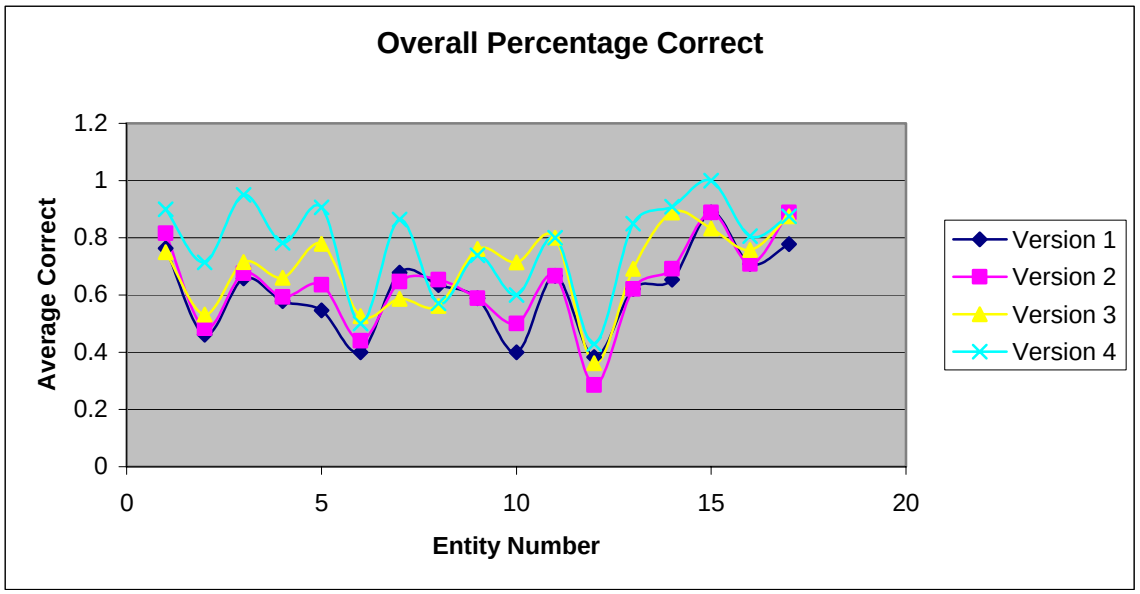

Figure 6. Chart Displaying Overall Intrinsic Evaluation Results.

[8] This accounts for almost 15% of the final output.

B. INTERPRETATION OF RESULTS

As can be seen in Figure 6, each of the four versions experienced most of the same peaks and troughs, meaning that for the most part, each version had similar difficulties and triumphs. However, with each iteration of the system, the performance curve shifted upwards, resulting in version four achieving the best results overall. The individual results of each version and the peculiarities observed will now be discussed.

1. Version 1

As detailed in Chapter III, version 1 of the system simply used term frequency to determine the person's organization, title, and location. Term frequency is a general measure of how often a term is found in a collection of documents. It is widely utilized in information retrieval, most often to provide a weight for a certain term given a certain document. "There are several variants, but a common form of the governing equation is:

$$w_{ij} = tf_{ij} * \log_2 \frac{N}{n} \qquad (3)$$

where w_{ij} is the weight of term t_i in document d_j, tf_{ij} is the frequency of term t_i in document d_j, N is the number of documents in the corpus, and n is the number of documents in the corpus in which term t_i occurs" (Mani et al., 01). In this research, the actual weights were not needed, but the frequency of each title, organization, and location in each entity's block was a necessary piece of information to make a decision.

Moreover, a blind test for the presence of certain cue words like "meet" and "brother" triggered the system to extract the sentence and classify it as either associate or familial information respectively. The span of the results seen stretch from an 88.9% high to a 38.3% low. The mean score was 61.2% while overall system precision *P* was calculated as follows:

$$P = \frac{\text{number of total correct inclusions}}{\text{total number of inclusions}} = 58.9\%$$

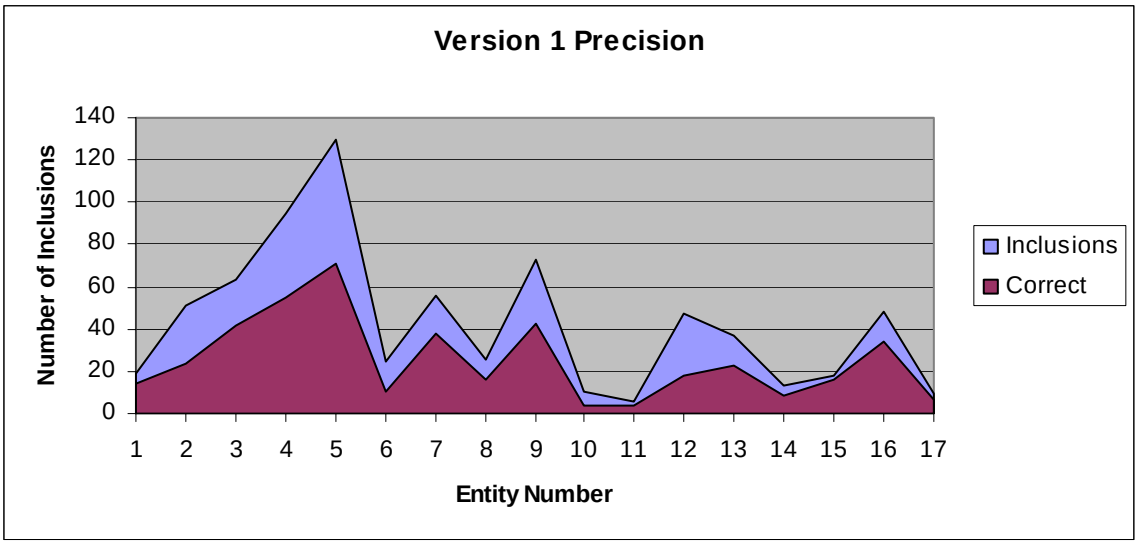

Figure 7. Precision of version 1.

The results from version 1 spurred more research questions. The system decided country and city independently to calculate the location, yet the system decided the city correctly less than 6% of the time. Meanwhile, the correct country was chosen 76% of the time. What was causing the discrepancy?

Also, the filters used to clue into familial relationships in the text were not working as evidenced by only a 38% success rate. The mystery of these results was clearly unraveled, however, by noticing that one person, Muhammad Ali Jinnah, was known as the "Father of the Nation of Pakistan." So, though he was mentioned as such 44 times, only 15 times was his actual family mentioned. Without his inclusion, the familial relationship filter of version 1 operated at approximately 46%.

2. Version 2

Version 2 of the system used term frequency with word ordering to determine organization, title, and location. Improvements were implemented in the familial relationship filter so that simple phrasal constructs such as "son of" and "father of" would be recognized. For version 2, the results spanned from 88.9% to 28.6%. The mean precision was 63.5% and the overall performance of the system was 59.2%. The biggest improvement of the system in this version was the taming of the city list which was found

to contain many different location errors, such as locations that did not actually exist. How this came to be is unknown, as these locations did not appear in the original PakNews corpus. After hand correcting the list, the overall precision of the city decision algorithm was 35.3%. The familial filter still struggled, however, and actually decreased in performance to 35.3%. This was caused by the inclusion of more information by the filter even though less of what was captured was accurate about the person of interest.

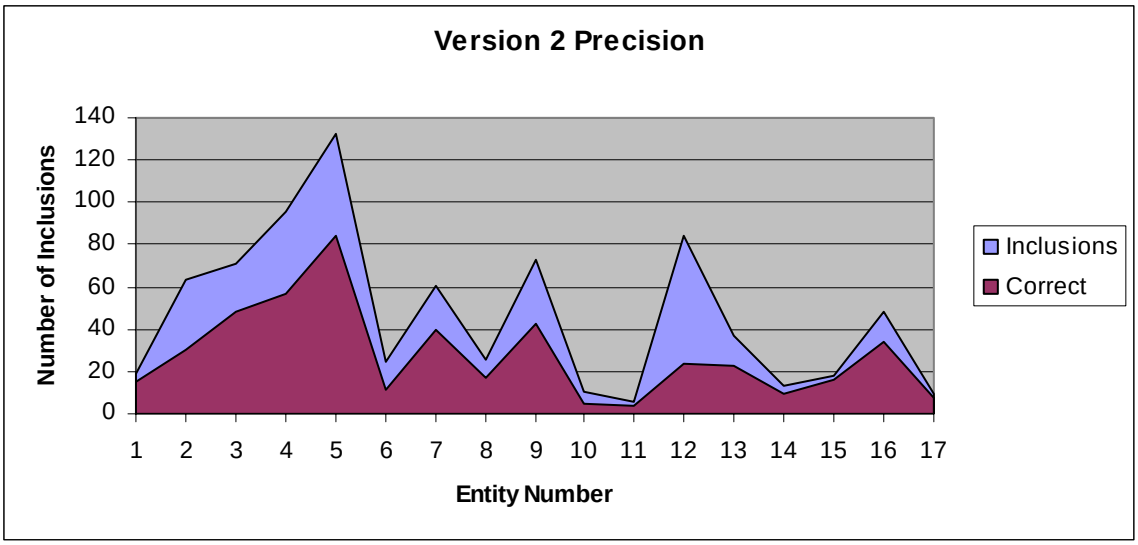

Figure 8. Precision of version 2.

3. Version 3

The third iteration of the system continued to perform better than the first two. The largest improvement for this version was the actual person names and job titles. Previous best for these two categories was 97% and 78% respectively. Version three pushed the name precision to 100% [9] and increased the title precision to 93.8%.

[9] This should come as no surprise to the reader, since this is essentially given information from the entity disambiguation system.

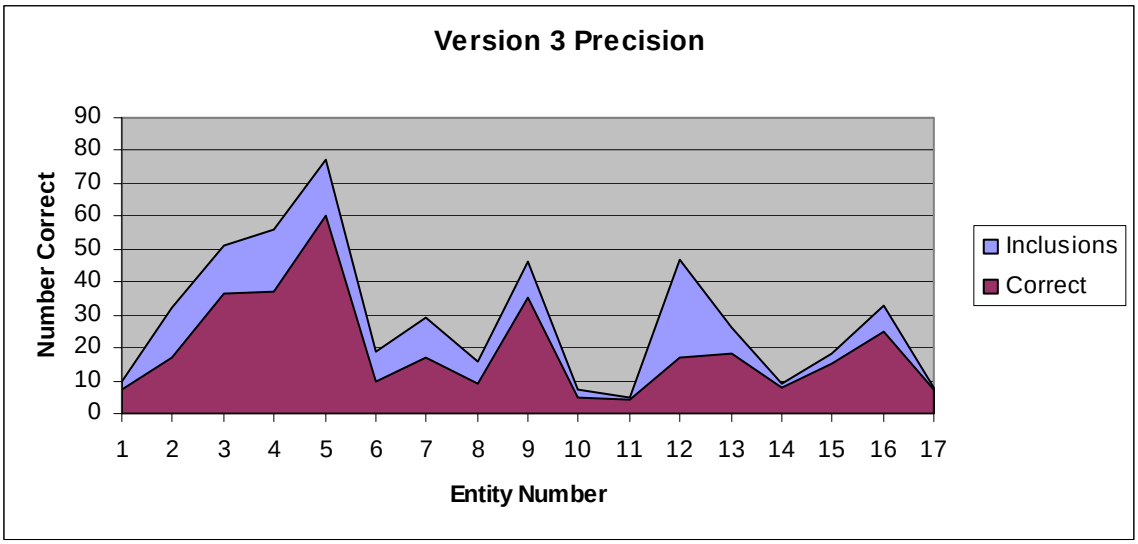

Figure 9. Precision of version 3.

The graph of the performance of version 3 (Figure 9) does not look very dissimilar from version 2. In fact version 3's overall performance was 67%, just an 8% increase from version 2. Version 3's results spanned from 36.2% to 88.9% with a mean precision of 69.4%. Deciding an entity's location continued to be a problem. In version 2, the system's precision was fairly high for deciding countries and extremely low for deciding cities. Version 3's new method for determining location flattened the scores of each category and caused them both to average out around 50%.

4. Version 4

Version 4 scored the best of all of the versions and achieved a better score in all categories except for title and organization. However, since these filters were developed in a modular way, the better performing filters from version 3 could easily be "plugged in" to version 4. The performance impact on the overall system's precision was not statistically significant. What was significant was that ignoring vernacular phrases in the familial filter caused the precision of that filter to increase from 37% to 55%. Also significant was the increase in the precision of the location filters. The city filter

exceeded 70% and the country filter exceeded 80%. The graph of the precision of this system shows that the balance between what to include versus what was correct about the entities was almost attained.

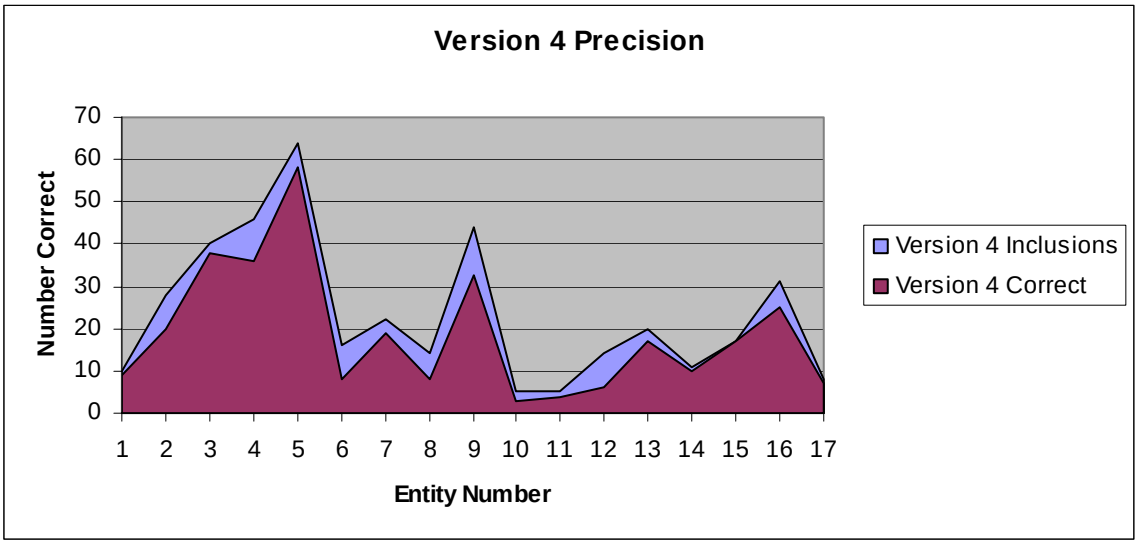

Figure 10. Precision of version 4.

The span for version 4's results went from 42.8% to 100% and achieved a mean precision of 77.6%. The overall system performed at 80.4%, a 13% increase over version 3. Due to the very high ratio of inclusions to correct inclusions, it was decided that any additional improvements to the system would benefit in only minimal gains and the limits of keyword filtering had probably been reached.

THIS PAGE INTENTIONALLY LEFT BLANK

V. CONCLUSIONS AND FUTURE WORK

A. CONCLUSIONS

In this thesis, the implied standing assumption that a sufficient biography summarization system would have to be equipped with the latest natural language processing tools was challenged. Just as automatic summarization is a divergent branch from the main trunk of natural language processing research, so biography summarization is sufficiently different from regular summarization to warrant a fresh look at techniques being used for the research being done. In general, it is not sufficient to simply use rigid summarization approaches for biography summarization.

Several lessons were learned from this research. First, even in a corpus as vast as PakNews, the large majority of entities will only be mentioned once. This shortage of final data complicates evaluation. However, several summarization projects were limited to a corpus of between 100 and 200 documents, so in this light the number of persons is adequate.

Second, providing regular expressions that will fit every case contained in a corpus while attempting to preserve generality is difficult. Several expressions went through iterations of change as small subtleties in how they were being used came to light. For example, placing a space after the given first name of an entity made all the difference between selecting the name “Abdul” who was a Sheikh and “Abdullah” who was a Prince.

Third, a statistical keyword filtering approach is sufficient enough to reach 80% correctness (i.e., precision). While a semantic parsing module may have taken the performance to near 100%, the amount of work required to achieve accurate parsing may not always be justified in every situation. If a system was being developed for the DoD to perform the same type of work, it would need to be close to 100% accurate, so it would be prudent to implement a semantic parsing algorithm.

B. FUTURE WORK

While the basic goal of producing biographical summarizations was achieved, there were many subtleties of the research that did not come to fruition. First, it would have been desirable to produce a version of the summarization system that *did* rely on semantic interpretation of the text, in order to quantitatively show how much value this added in the final evaluation. The immediate obstacle for dealing with semantics was that the PakNews corpus had not previously been tagged with part-of-speech tags.

Second, more filters could have been developed to deal with several other different types of information. For example, there have been recent methods detailed to extract and interpret temporal information (Bethard et al., 07). Given such information, a very robust timeline for the person could have been created and would have allowed for the determination of if the person was still living, when different family would have died, etc. When working with temporal information, inference is also key to understanding the text.

It would have also been profitable to perfect the nationality and lifespan filters. The major problem with these filters was sparse data from the corpus. Yet, it is possible that this would be expected since the news articles being processed are neither wholly about one person, nor are they encyclopedia articles about the person. The temporal filtering described above could have aided the lifespan filter inasmuch as knowing the earliest date an entity is mentioned and extrapolating their birth as being before that date.

Third, the output produced could be useful in social networking applications and query-based applications. A front-end interface for the system was envisioned which would parse the XML output and present the user with the concise facts regarding a person. A map could be displayed based on the location information gathered. In fact, a map showing all locations that an entity is mentioned could also be useful as a pictorial way to trace a person's movements. In addition to the mapping possibilities, the social networking visualization possible is substantial. Named associates of the current person can be placed on a graph or grid with each associate becoming a node. User's could then click nodes and navigate through a chain of persons to reveal degrees of separation

between two persons. Finally, the XML backend data driving the interface could be completely indexed for quick data retrieval and to facilitate question answering.

In the area of evaluation, several additional approaches could still be addressed. An evaluation could be performed over the difference between the time necessary to generate the biography by hand versus by machine. The hand generated summaries could also prove as a basis to score the generated biography for correctness and informativeness. In addition to these relatively subjective evaluations, an evaluation could also be conducted over each version of the system based upon sentence recall. It is a statistical fact that precision can be increased at the expense of recall and vice versa. A full evaluation of the 18 entities examined in the current evaluation needs to be undertaken to see how recall was impacted by the decreasing information inclusions.

In conclusion, the topic of automatic biographical summarization is a fascinating one that is both relevant and challenging in our current age. In addition to the scientific research that must still be conducted to determine methods to create nearly perfect biographies autonomously, philosophical issues of privacy and individual rights must be addressed as well. The current growth of information online that is publicly available and available for sale is leading our culture to an ever-increasing demolition of individual secrecy. Systems like the one described in this research must be developed with the safeguards of a will to use the information gleaned for mankind's benefit and not for personal gain.

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

(Alani et al., 03) Alani, Harith, Sanghee Kim, David E. Millard, and Mark J. Weal. "Automatic Ontology-Based Knowledge Extraction and Tailored Biography Generation From the Web." IEEE Intelligent Systems 18 (2003): 14-21.

(Aone et al., 99) Aone, C, M E. Okurowski, J Gorlinsky, and B Larsen. "A Trainable Summarizer with Knowledge Acquired From Robust NLP Techniques." Advances in Automatic Text Summarization. Ed. Inderjeet Mani and Mark T. Maybury. Cambridge, Massachusetts: The MIT Press, 2001. 71-81.

(Barzilay, 03) Barzilay, Regina. "Summarization." MIT. 6.892: Computational Models of Discourse. MIT, Cambridge, MA. 8 Mar. 2003. Feb. 2007 <http://people.csail.mit.edu/regina/6892/lec10/lec10.pdf>.

(Bethard et al., 07) Bethard, Steven, James H. Martin, and Sara Klingenstein. First IEEE International Conference on Semantic Computing, 17 Sept. 2007, IEEE Computer Society. July 2007 <http://ucsu. colorado.edu/~bethard/documents/bethard_07_syntactic_ temporal.pdf>.

(Blume, 05) Blume, Matthias. International Conference on Intelligence Analysis, 2005. Sept. 2006 <https://analysis.mitre.org/proceedings/ Final_Papers_Files/12_Camera_Ready_Paper.pdf>.

(Earl, 70) Earl, L. L. Experiments in Automatic Extracting and Indexing. Information Storage and Retrieval, 6(6); 313 – 334; 1970.

(Edmundson, 69) Edmundson, H P. "New Methods in Auotmatic Extracting." Journal of the Association for Computing Machinery 16 (1969): 264-285.

(Feng et al, 06) Feng, D. and Ravichandran, D. and Hovy, E. 2006. Mining and Re-ranking for Answering Biographical Queries on the Web. In *Proceedings of the Twenty-First National Conference on Artificial Intelligence*,2-8. Menlo Park, Calif.:AAAI Press.

(Kupiec et al., 95) Kupiec, J, J Pedersen, and F Chen. *Proceedings of the 18th ACM-SIGR Conference*. Annual ACM Conference on Research and Development in Information Retrieval, 1995, ACM. Seattle, Washington: ACM, 1995.

(Jurafsky et al., 01) Jurafsky, Daniel, and James H. Martin. Speech and Language Processing: an Introduction to Natural Language Processing, Computational Linguistics and Speech Recognition. 2nd ed. Boston, MA: Prentice Hall, 2001.

(Luhn, 58) Luhn, H P. "The Automatic Creation of Literature Abstracts." IBM Journal of Research & Development 2 (1958): 159-165.

(Mani, 01) Mani, Inderjeet. Summarization Evaluation: An Overview. Proceedings of the Second NTCIR Workshop on Research in Chinese & Japanese Text Retrieval and Text Summarization, Mar. 2001, National Institute of Informatics. 1 Mar. 2007 <http://research.nii.ac.jp/ntcir/workshop/OnlineProceedings2/sum-mani.pdf>.

(Mani et al., 01) Mani, Inderjeet, and Mark T. Maybury. Advances in Automatic Text Summarization. Cambridge, MA: MIT P, 2001.

(McKeown et al., 95) McKeown, K., J. Robin, and K. Kukich. "Generating Concise Natural Language Summaries." Information Processing and Management 31.5 (1995): 703-733.

(Miller et al., 90) Miller, George A. Proceedings of the 1994 Human Language Technology Workshop. 1994. 19 Sept. 2007 <http://acl.ldc.upenn.edu/H/H94/H94-1111.pdf>.

(Morris et al., 92) Morris, A. H., G. M. Kasper, and D. A. Adams. "The Effects and Limitations of Automated Text Condensing on Reading Comprehension Performance." Information Systems Research 3.1 (1992): 17-35.

(Myaeng, 96) Myaeng, S H., and D Jang. "Development and Evaluation of a Statistically Based Document Summarization System." Advances in Automatic Text Summarization. Ed. Inderjeet Mani and Mark T. Maybury. Cambridge, Massachusetts: The MIT Press, 2001. 61-70.

(Paice, 89) Paice, Chris D. "Constructing Literature Abstracts by Computer: Techniques and Prospects." Information Processing and Management: an International Journal 26 (1990): 171-186.

(Saggion, 04) Saggion, Horacio. "Automatic Text Summarization: Past, Present, and Future." University of Sheffield, England. 2004. Feb. 2007 <http://www.dcs.shef.ac.uk/~saggion/saggion04.PDF>.

(Schiffman et al., 01) Schiffman, Barry, Inderjeet Mani, and Kristian J. Concepcion. Proceedings of the 39th Annual Meeting on Association for Computational Linguistics. Annual Meeting of the ACL, Association for Computational Linguistics. Morristown, NJ: Association for Computational Linguistics, 2001.

(Smiraglia, 02) Smiraglia, Richard P. Works as Entities for Information Retrieval. New York, NY: Haworth P, 2002. Google Book Search. July 2007 <http://books.google.com/books?id=KGE_0WfZaCEC&dq=&sa=X&oi=print&ct=book-ref-page-link>.

(Sparck Jones, 97) Sparck Jones, K. "Automatic Summarizing: Factors and Directions." Advances in Automatic Text Summarization. Comp. Inderjeet Mani and Mark T. Maybury. Cambridge, MA: MIT Press, 2001. 1-12.

(Zhou, 05) Zhou, Liang, Miruna Ticrea, and Eduard Hovy. Proceedings of EMNLP. Conference on Empirical Methods in Natural Language Processing, 25 July 2004, Association for Computational Linguistics. Barcelona, Spain: EMNLP, 2004.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX A – FAIR ISAAC INPUT

A. TITLES

Academician Doctor
Admiral
Admiral Sir
Advisor
Advisor Doctor
Advisor Finance Doctor
Advisor Governor
Advisor Governor Home Secretary
Advisor Sayyid
Agriculture
Agriculture Minister
Ambassador
Ambassador Doctor
Ambassador Doctor Ms
Ambassador Finance Minister
Ambassador General
Ambassador Madam
Ambassador Major General
Ambassador Maulvi
Ambassador Mian
Ambassador Mr
Ambassador Mrs
Ambassador Ms
Ambassador Mullah
Ambassador Sayyid
Ambassador Sir
Ambassador Spokesman

Assistant Attorney General
Assistant Coach
Assistant Commissioner
Assistant Commissioner Mr
Assistant Defense Secretary
Assistant Director
Assistant Director Captain
Assistant Foreign Minister
Assistant Minority Leader
Assistant Professor
Assistant Secretary
Assistant Secretary General
Assistant Secretary Ms
Assistant Technical Officer
Attache
Attache Colonel
Attache Commander
Attorney
Attorney General
Attorney General Honorable
Ayatollah
Baroness
Barrister
Barrister Doctor
Barrister Doctor Justice Qazi
Barrister Mrs
Barrister President
Barrister Sayyid
Barrister Sultan
Barrister Sultan Prime Minister
Baseman
Batsman

Batsman Bowler
Batsman Captain
Batsman Skipper
Bishop
Bowler
Brigadier
Brigadier Chief Executive
Brigadier Doctor
Brigadier General
Brigadier General Sayyid
Brigadier General Sheikh
Brigadier Mian
Brigadier Mrs
Brigadier Retired
Brigadier Sayyid
Brigadier Shaheed
Brigadier Sultan
CEO
Caliph
Captain
Captain Consul Officer
Captain Doctor
Captain President Premier
Captain Skipper
Cardinal
Chairman Ambassador
Chairman Congressman
Chairman Director
Chairman Doctor
Chairman General
Chairman Haji
Chairman Lieutenant General

Chairman Lord
Chairman Major
Chairman Major General
Chairman Maulana
Chairman Mian
Chairman Mian President
Chairman Mr
Chairman Mrs
Chairman Ms
Chairman Premier
Chairman President
Chairman Professor
Chairman Sayyid
Chairman Secretary
Chairman Secretary General
Chairman Senator
Chairman Senator Professor
Chairman Sheikh
Chairwoman
Chairwoman Mr
Chairwoman Prime Minister Senator
Chancellor
Chancellor Lord
Chancellor Mr
Chaplain
Chaplain Captain
Chief
Chief Admiral
Chief Brigadier
Chief Captain
Chief Doctor
Chief Executive

Chief Executive Brigadier
Chief Executive Doctor
Chief Executive General
Chief Executive General President
Chief Executive Major General
Chief Executive Mr
Chief Executive Officer
Chief Executive Officer Mr
Chief Executive President
Chief Executive President General
Chief General
Chief Justice
Chief Justice Judge
Chief Justice Justice
Chief Justice Mian
Chief Justice Sayyid
Chief Justice Sheikh
Chief Lieutenant General
Chief Lieutenant General Retired
Chief Marshal
Chief Marshal President
Chief Marshal Sir
Chief Maulana
Chief Mian
Chief Minister
Chief Minister Doctor
Chief Minister Governor
Chief Minister Maharaja
Chief Minister Maharaja Captain
Chief Minister Mian
Chief Minister Mr
Chief Minister Sayyid

Chief Mr
Chief Officer
Chief Petty Officer
Chief President Mian
Chief Qazi
Chief Sayyid
Chief Secretary
Chief Secretary Doctor
Chief Secretary Governor
Chief Secretary Mian
Chief Secretary Mr
Chief Secretary Prime Minister
Chief Secretary Speaker
Chief Shaheed
Chief Sheikh
Chief Whip
Class
Coach
Colonel
Colonel Commandant Lieutenant General
Colonel Doctor
Colonel General
Colonel Haji
Colonel President
Colonel Retired
Colonel Sayyid
Colonel Sultan
Command Lieutenant General
Commandant
Commandant Chief Secretary
Commandant Officer President
Commander

Commander Admiral
Commander Brigadier
Commander Colonel
Commander Director
Commander General
Commander Governor
Commander Haji
Commander Haji Hazrat
Commander Hazrat
Commander Lieutenant Colonel
Commander Lieutenant General
Commander Lieutenant General Captain
Commander Lieutenant General Sayyid
Commander Major
Commander Major General
Commander Maulvi
Commander Mr
Commander Mullah
Commander Rear Admiral
Commander Sayyid
Commander Shaheed
Commander Vice Admiral
Commerce Minister
Commerce Minister Mr
Commerce Minister Sayyid
Commerce Secretary
Commissioner
Commissioner Doctor
Commissioner Justice
Commissioner Justice Retired
Commissioner Mr
Commissioner Mrs

Commissioner Ms
Commissioner Qazi
Commissioner Retired Chief Justice
Commissioner Sayyid
Commissioner Sheikh
Commissioner Sir
Commodore
Commodore Sayyid
Comrade
Congressman
Congressman Congressman
Congressman Representative
Congresswoman
Congresswoman Representative
Constable
Consul
Consul General
Consul General Haji
Consul Officer
Corporal
Councilman
Councilman Mr
Councilor
Councilor Haji
Councilor Qazi
Counselor
Defender
Defense
Defense Attache Brigadier
Defense Attache Colonel
Defense Minister
Defense Minister Admiral

Defense Minister Colonel General
Defense Minister General
Defense Minister Madam
Defense Minister Ms
Defense Minister Prince
Defense Minister Rear Admiral
Defense Minister Sheikh
Defense Mr
Defense Sayyid
Defense Secretary
Defense Secretary Lieutenant General
Delegate
Delegate Brigadier
Delegate Mr
Delegate Ms
Delegate Mullah
Delegate Sayyid
Deputy
Deputy Ambassador
Deputy Attorney
Deputy Attorney General
Deputy Attorney General Barrister Sultan
Deputy Attorney General Sayyid
Deputy Chairman
Deputy Chairman Mr
Deputy Chief
Deputy Chief Colonel
Deputy Chief Executive
Deputy Chief Minister
Deputy Commandant Sayyid
Deputy Commander
Deputy Commissioner

Deputy Defense Minister
Deputy Defense Minister General
Deputy Defense Secretary
Deputy Director
Deputy Director General
Deputy District Health Officer Doctor
Deputy District Officer Health Doctor
Deputy Executive Director
Deputy Foreign Minister
Deputy Foreign Minister Mr
Deputy Foreign Minister Mrs
Deputy Foreign Minister Ms
Deputy Foreign Minister Mullah
Deputy Foreign Secretary Maulvi
Deputy Governor
Deputy Information Minister
Deputy Interior Minister Prince
Deputy Mayor
Deputy Minister
Deputy National Security Advisor
Deputy Premier Prince
Deputy President
Deputy Prime Minister
Deputy Prime Minister Doctor
Deputy Prime Minister Haji
Deputy Prime Minister Mr
Deputy Prime Minister President
Deputy Prime Minister Sheikh
Deputy Prime Minister Sultan
Deputy Secretary
Deputy Secretary General
Deputy Secretary General Senator

Deputy Secretary Mr
Deputy Skipper
Deputy Speaker
Deputy Speaker Governor
Deputy Speaker Haji
Deputy Speaker Sayyid
Deputy Spokesman
Deputy Superintendent
Deputy Superintendent Chairman
Deputy Treasury Secretary
Director
Director Admiral
Director Ambassador
Director Brigadier
Director Chairman
Director Doctor
Director General
Director General Doctor
Director General Health Doctor
Director General Lieutenant General
Director General Major General
Director General Mr
Director General Sayyid
Director Lieutenant General
Director Major General
Director Mr
Director Mr Doctor
Director Mr Sayyid
Director Mrs
Director President Mr
Director Qazi
Director Sayyid

District Attorney
District General Secretary
District Health Officer
District Health Officer Doctor
District President
District President Haji
Division
Division Deputy Secretary
Division Doctor
Division Lieutenant General
Division Mr
Doc
Doctor
Doctor Ambassador
Doctor Ambassador Ms
Doctor Barrister
Doctor Captain
Doctor Commander
Doctor Doctor
Doctor Foreign Minister
Doctor Governor
Doctor Maulana
Doctor Mian
Doctor Minister
Doctor Mr
Doctor Mrs
Doctor Nawab
Doctor President
Doctor Privatization Minister
Doctor Professor
Doctor Professor Technology Minister
Doctor Qazi

Doctor Sayyid
Doctor Secretary
Doctor Senator
Doctor Sheikh
Doctor Sheikh Sultan
Doctor Sultan
Doctor Technology Professor
Doctor Vice President Sayyid
Elder
Emeritus Doctor
Emperor
Executive
Executive Brigadier
Executive Director
Executive Director Doctor
Executive General
Executive Officer
Executive Secretary
Father
Fielder
Finance Minister
Finance Minister Doctor
Finance Minister Mr
Finance Minister Sayyid
Finance Minister Senator
Finance Secretary
Finance Secretary Doctor
Finance Senator
Financial Advisor
Financial Minister
First Deputy Finance Minister
First Deputy Prime Minister Sheikh

First Lady
First Lady Mrs
First Lady Senator
First President
First President General
First Secretary
First Secretary Mr
First Vice President
First Vice President Doctor
Foreign Minister
Foreign Minister Doctor
Foreign Minister Maulvi
Foreign Minister Mian
Foreign Minister Mr
Foreign Minister Mrs
Foreign Minister Ms
Foreign Minister Mullah
Foreign Minister Prince
Foreign Minister Sayyid
Foreign Minister Secretary
Foreign Minister Sheikh
Foreign Secretary
Foreign Secretary Mr
Foreign Secretary Sayyid
Gen
General
General Colonel
General Defense Minister
General Doctor
General Foreign Secretary
General Maulvi
General Minister

General Mr
General Officer
General President
General Retired
General Sayyid
General Secretary
General Secretary Chairman
General Secretary Haji
General Secretary Mian
General Senator
General Sheikh
General Sir
General Speaker
General Staff General
General Staff Lieutenant General
General Staff Major General
Goalkeeper
Governor
Governor Chief Minister
Governor Chief Secretary
Governor Doctor
Governor Finance Minister
Governor General
Governor Haji
Governor Home Chief Minister
Governor Honorable
Governor Justice
Governor Lieutenant General
Governor Lieutenant General Sayyid
Governor Mian
Governor Mr
Governor Mullah

Governor Prince
Governor Retired Lieutenant General
Governor Sayyid
Grand Ayatollah
Haji
Haji Hazrat
Haji Mullah
Haji Nawab
Haji President
Haji Qazi
Haji Sayyid
Hazrat
Hazrat Imam
Hazrat Imam Colonel
Hazrat Justice Mian
Hazrat Mr
Hazrat Prime Minister
Hazrat Prophet
Hazrat Senator
Hazrat Sheikh
Health Doctor
Health Major General
Health Minister
Health Minister Doctor
Health Minister Mr
Health Minister Mullah
Health Minister Professor Doctor
Health Mr
Health Mrs
Health Secretary
Health Secretary Doctor
Home Minister

Home Minister Mr
Home Mr
Home Secretary
Home Secretary Brigadier
Home Secretary Doctor
Home Secretary Governor
Home Secretary Sayyid
Honorable
Honorable Barrister Sultan
Honorable General
Honorable Minister
Honorable President
Imam
Imam Hazrat
Imam Sayyid
Imam Senator
Imam Sheikh
Information Director Mrs
Information Doctor
Information Minister
Information Minister Doctor
Information Minister Mr
Information Minister Mr Sheikh
Information Minister Sayyid
Information Minister Senator
Information Minister Sheikh
Information Mr
Information Mrs
Information Officer
Information Secretary
Information Secretary Sayyid
Information Secretary Sayyid Chairman

Information Technology Barrister
Information Technology Minister
Information Technology Mr
Interim Foreign Minister
Interim Foreign Minister Doctor
Interim Leader
Interim Leader President
Interim Prime Minister
Interior Minister
Interior Minister Doctor
Interior Minister General
Interior Minister Lieutenant General
Interior Minister Major General
Interior Minister Ms
Interior Minister Prince
Interior Minister Sayyid
Interior Minister Sheikh
Interior Secretary
Judge
Judge Justice
Judge Ms
Judge Sayyid
Judge Sheikh
Junior
Junior Home Minister
Justice
Justice Barrister
Justice Doctor
Justice Judge
Justice Mian
Justice Minister
Justice Minister Mullah

Justice Mrs
Justice Qazi
Justice Sayyid
Justice Sheikh
Justice Sheikh Qazi
Keeper
Keeper Batsman
Keeper Skipper
Lady
Lance Corporal
Leader Barrister Sultan
Leader Colonel
Leader Congresswoman
Leader Doctor
Leader Maulana
Leader Maulvi
Leader Mian
Leader Mr
Leader Mullah
Leader Qazi
Leader Sayyid
Leader Sheikh
Lieutenant
Lieutenant Colonel
Lieutenant Colonel Sayyid
Lieutenant Commander
Lieutenant Doctor
Lieutenant General
Lieutenant General Chairman
Lieutenant General Commander
Lieutenant General General
Lieutenant General Retired

Lieutenant General Retired Sayyid
Lieutenant General Sayyid
Lieutenant General Sir
Lieutenant Governor
Lord
Lord Mayor
Lord Professor
Lord Sir
Madam
Madam Vice President
Madame
Maestro
Maharaja
Maharaja Ms
Major
Major Director General
Major Doctor
Major General
Major General Brigadier
Major General General
Major General Haji
Major General Imam
Major General Imam Mr
Major General Nawab
Major General Sayyid
Major General Sheikh
Major General Sir
Major Sayyid
Marshal Sayyid
Master General
Master General Lieutenant General
Master Sergeant

Maulana
Maulana Doctor
Maulana Haji
Maulana Maulvi
Maulana President
Maulana Professor
Maulana Qazi
Maulana Sayyid
Maulana Secretary General
Maulana Senior Minister
Maulana Sultan
Maulvi
Maulvi Maulana
Maulvi Sayyid
Mayor
Medical Officer
Medical Officer Doctor
Medical Superintendent
Medical Superintendent Doctor
Medical Superintendent Lieutenant Colonel
Mian
Mian Chief Minister Mr
Mian Doctor
Mian Foreign Minister
Mian Governor
Mian Officer Sayyid
Mian Prime Minister
Mian Sayyid
Mian Secretary General
Midfielder
Minister
Minister Baroness

Minister Barrister
Minister Barrister Sultan
Minister Brigadier
Minister Chief Secretary
Minister Defense
Minister Doctor
Minister Doctor Sayyid
Minister General
Minister Haji
Minister Health Doctor
Minister Lieutenant General
Minister Lord
Minister Maulana
Minister Mian
Minister Mr
Minister Mrs
Minister Ms
Minister Mullah
Minister Nawab
Minister Premier
Minister President
Minister Prince
Minister Sayyid
Minister Senator
Minister Sheikh
Minister Spokesman
Minority Leader
Miss
Mr
Mr Ambassador
Mr Assistant Foreign Minister
Mr Assistant Secretary

Mr Chairman
Mr Chairman Mian
Mr Chief Minister
Mr Colonel General
Mr Commander
Mr Commander Mullah
Mr Commerce Minister
Mr Commissioner
Mr Congressman
Mr Consul General
Mr Defense Minister
Mr Defense Secretary
Mr Deputy Chairman
Mr Deputy Chief
Mr Deputy Commissioner
Mr Deputy National Security Advisor
Mr Deputy Prime Minister
Mr Deputy Spokesman
Mr Director
Mr Director General
Mr Doctor
Mr Executive Director
Mr Finance Minister
Mr Foreign Minister
Mr Foreign Minister Mian
Mr Foreign Secretary
Mr General
Mr General President
Mr Haji
Mr Hazrat
Mr Health Minister Doctor
Mr Home Minister

Mr Home Secretary
Mr Information Minister
Mr Interior Minister
Mr Judge
Mr Justice
Mr Justice Mian
Mr Justice Qazi
Mr Justice Sayyid
Mr Justice Sheikh
Mr Lieutenant General
Mr Lord
Mr Major General
Mr Maulana
Mr Medical Superintendent Doctor
Mr Minister
Mr Mrs
Mr Mullah
Mr Nawab
Mr Premier
Mr President
Mr President General
Mr Prime Minister
Mr Prince
Mr Privatization Senator Doctor
Mr Professor
Mr Qazi
Mr Rear Admiral
Mr Sayyid
Mr Second Secretary
Mr Secretary
Mr Secretary General
Mr Secretary General Professor

Mr Senator
Mr Sheikh
Mr Speaker
Mr Spokesman
Mr Sultan
Mr Telecommunications Minister
Mr Trade Minister
Mr Treasury Secretary
Mr Vice President
Mrs
Mrs Commissioner
Mrs Doctor
Mrs Major
Mrs Minister Ms
Mrs Ms
Mrs President
Mrs Prime Minister
Mrs Sayyid
Ms
Ms Ambassador
Ms Doctor
Ms Executive Director
Ms Information Minister
Ms Judge
Ms Justice
Ms Maharaja
Ms Mrs
Ms President
Ms Senator
Ms Vice President
Mujahid
Mujahid Commander

Mullah
Mullah Haji
Mullah President
National Security Advisor
National Security Advisor Doctor
National Security Advisor Mr
Nawab
Nawab Sayyid
Officer
Officer Assistant Director
Officer Brigadier
Officer Colonel
Officer Deputy Director
Officer Director
Officer Doctor
Officer General
Officer Lieutenant Colonel
Officer Mian
Officer Mr
Officer Ms
Officer Qazi
Officer Sayyid
Ombudsman
Ombudsman Justice
Opener
PM
Pace Bowler
Pace-Bowler
Paceman
Parliamentarian
Parliamentarian Chief
Parliamentarian President

Parliamentarian Sheikh
Pastor
Patriarch
Petroleum Minister
Pitcher
Pope
Premier
Premier Barrister Sultan
Premier Doctor
Premier Maulana
Premier Mr
Premier Prime Minister
Premier Vice Prime Minister
President
President Advisor
President Captain
President Chairman
President Chief Executive General
President Colonel
President Doctor
President Doctor Sayyid
President Executive General
President General
President Haji
President Justice
President Madam
President Major General
President Major General Retired
President Maulana
President Mian
President Mr
President Mrs

President Ms
President National
President Nawab
President Officer Director
President President
President Prime Minister
President Prince
President Professor
President Qazi
President Retired Major General
President Reverend
President Sayyid
President Secretary
President Senator
President Sheikh
President Sir
President Staff General
President Sultan
Prime Minister
Prime Minister Barrister Sultan
Prime Minister Chief Secretary
Prime Minister Doctor
Prime Minister General
Prime Minister Mian
Prime Minister Mr
Prime Minister Mrs
Prime Minister Ms
Prime Minister Premier
Prime Minister President
Prime Minister Professor
Prime Minister Sayyid
Prime Minister Secretary

Prime Minister Sheikh
Prime Minister Sultan
Prime Minister Technology Doctor
Prince
Prince Sheikh
Prince Sultan
Princess
Private
Private First Class
Privatization Doctor
Privatization Minister
Privatization Minister Doctor
Privatization Mr
Privatization Senator Doctor
Prof
Professor
Professor Chairman
Professor Doctor
Professor Doctor Mrs
Professor Doctor Sayyid
Professor Doctor Sheikh
Professor Doctor Technology Minister
Professor Maulana
Professor Mian Sayyid
Professor Minister Doctor
Professor Mr
Professor President
Professor Retired Doctor
Professor Sayyid
Professor Sultan
Professor Technology Doctor
Prophet

Prophet Hazrat
Provincial Agriculture Minister
Provincial Finance Minister
Provincial Finance Minister Sayyid
Provincial Health Minister
Provincial Health Minister Doctor
Provincial Minister
Provincial Minister Doctor
Provincial Minister Mian
Provincial Minister Sayyid
Qazi
Qazi Commissioner
Qazi Deputy
Qazi General
Qazi Justice
Qazi Maulana
Qazi Mrs
Queen
Queen Baroness
Rabbi
Rear Admiral
Rear Vice Admiral
Registrar
Representative
Representative Ambassador
Representative Congressman
Representative Congresswoman
Representative Doctor
Representative Mr
Retired Brigadier
Retired General
Retired Justice

Retired Lieutenant General
Retired Major
Retired Major General
Reverend
Reverend Doctor
Reverend Rabbi
Rifleman
Right Minister
Right Reverend
Right Reverend Doctor
Right Winger
Saint
Saint Hazrat
Sayyid
Sayyid Chairman
Sayyid Chief Minister
Sayyid Deputy Director
Sayyid Doctor
Sayyid General
Sayyid Governor
Sayyid Haji
Sayyid Information Secretary
Sayyid Lieutenant General
Sayyid Minister
Sayyid Mr
Sayyid Officer
Sayyid President
Sayyid Sultan
Science Doctor
Science Minister
Science Officer
Second Secretary

Second Secretary Mr
Second Sergeant
Secretary
Secretary Agriculture Mr
Secretary Ambassador
Secretary Brigadier
Secretary Brigadier Retired
Secretary Chairman
Secretary Chief Minister
Secretary Colonel Retired
Secretary Defense
Secretary Defense Lieutenant General
Secretary Defense Rear Admiral
Secretary Defense Sir
Secretary Deputy Speaker Governor
Secretary Doctor
Secretary Finance Mr
Secretary General
Secretary General Director
Secretary General Doctor
Secretary General Finance Mr
Secretary General Lord
Secretary General Maulana
Secretary General Mian
Secretary General Mr
Secretary General Professor
Secretary General Sayyid
Secretary General Senator
Secretary General Sir
Secretary Governor
Secretary Haji
Secretary Health Doctor

Secretary Health Mr
Secretary Honorable
Secretary Information Mrs
Secretary Information Ms
Secretary Information Senator
Secretary Information Technology Brigadier
Secretary Interior Mr
Secretary Justice
Secretary Major General
Secretary Maulana
Secretary Mian
Secretary Mr
Secretary Ms
Secretary Petroleum
Secretary President
Secretary Professor
Secretary Qazi
Secretary Rear Admiral
Secretary Sayyid
Secretary Senator
Secretary Senior Vice Chairman
Secretary Sheikh
Secretary Spokesman
Secretary Squadron Leader
Secretary Treasurer
Security Advisor Mr
Security Brigadier
Security Director
Security Secretary
Senator
Senator Chairman
Senator Doctor

Senator Haji
Senator Honorable
Senator Justice
Senator Maulana
Senator Mian
Senator Minister Sayyid
Senator Mr
Senator Mrs
Senator Ms
Senator President
Senator Professor
Senator Sayyid
Senator Secretary General
Senior
Senior Advisor
Senior Colonel
Senior Defense Minister
Senior Deputy
Senior Deputy President
Senior Director Doctor
Senior Executive President
Senior Executive Vice President
Senior Judge Justice
Senior Judge Justice Sheikh
Senior Minister
Senior Minister Maulana
Senior Minister Mr
Senior Minister Sayyid
Senior Provincial Minister
Senior Provincial Minister Maulana
Senior Spokesman
Senior Vice Chairman

Senior Vice Foreign Minister
Senior Vice Foreign Minister Mr
Senior Vice Minister
Senior Vice President
Senior Vice President Doctor
Senior Vice President Haji
Senior Vice President Sheikh
Sergeant
Sergeant First Class
Sergeant Major
Serviceman
Shah Imam
Shaheed
Shaheed Captain
Shaheed Doctor
Sheikh
Sheikh Doctor
Sheikh Hazrat
Sheikh Justice
Sheikh Sultan
Sheriff
Shortstop
Sir
Sir Sayyid
Sir Staff General
Sister
Skipper
Skipper Captain
Solicitor General
Speaker
Speaker Mr
Speaker Sayyid

Spokesman
Spokesman Brigadier
Spokesman Brigadier General
Spokesman Captain
Spokesman Colonel
Spokesman Commander
Spokesman Doctor
Spokesman General
Spokesman Interior Secretary
Spokesman Lieutenant
Spokesman Lieutenant Colonel
Spokesman Major
Spokesman Major General
Spokesman Mr
Spokesman Mr Foreign Minister
Spokesman Mullah
Spokesman Professor
Spokesman Rear Admiral
Spokesman Sayyid
Spokesman Sheikh
Spokesman Sultan
Spokesperson
Spokesperson Major General
Spokeswoman
Spokeswoman Captain
Spokeswoman Major
Squadron Leader
Staff Admiral
Staff Admiral Sir
Staff General
Staff General Sir
Staff Lieutenant General

Staff Lieutenant General Sayyid
Staff Major General
Staff Sergeant
Staff Sir
Striker
Sultan
Sultan Mian
Sultan Prince
Superintendent
Supervisor
Supreme Commander Mullah
Supreme Leader Ayatollah
Supreme Leader Maulana
Supreme Leader Mullah
Technical President
Technology Advisor
Technology Doctor
Technology Lieutenant Colonel
Technology Minister
Technology Minister Doctor
Technology Minister Professor
Technology President Professor
Technology Prime Minister Professor
Technology Professor
Technology Professor Doctor
Telecommunications Minister
Telecommunications Mr
Terrorist
Trade Minister
Trade Minister Baroness
Trade Minister Mr
Trade Mr

Trade Mrs
Trade Representative
Trade Representative Ambassador
Trade Representative Mr
Treasurer
Treasurer Doctor
Treasury Secretary
Treasury Secretary Mr
Treasury Undersecretary
Umpire
Undersecretary
Vice Admiral
Vice Captain
Vice Chairman
Vice Chairman Mian
Vice Chairman Sayyid
Vice Chancellor
Vice Chancellor Professor Doctor
Vice Chief Brigadier
Vice Foreign Minister
Vice Foreign Minister Mr
Vice Marshal
Vice Minister
Vice Premier
Vice Premier Doctor
Vice Premier Mr
Vice President
Vice President Chairman
Vice President Doctor
Vice President Haji
Vice President Mian
Vice President Minister

Vice President Mr

Vice President Qazi

Vice President Sayyid

Vice President Sheikh

Vice Prime Minister

Wicketkeeper

Wicketkeeper-batsman

Winger

B. ORGANIZATIONS

accountability bureau|ab

accountability court|ac

aga khan development network|akdn

aga khan education service|akes

aga khan foundation|akf

aga khan rural support program|akrsp

aga khan university|aku

agricultural development bank|adb

agricultural linkages|agricultural linkage

agriculture department|agri department

agriculture development bank|adb

agriculture ministry|agri ministry

agriculture research|agri research

air chief marshal|acm

air chiefs|air chief

air forces|air force

al qaeda|aq

al qaida|aq

all china women federation|acwf

all india radio|air

all pakistan mohajir students organization|apmso

all pakistan newspapers employees confederation|apnec

all pakistan newspapers society|apns
all pakistan women association|apwa
all parities conference|apc
all parities hurriyat conference|aphc
all party conference|apc
all party hurriyat conference|aphc
all party hurriyet conference|aphc
allama iqbal open university|aiou
allied bank limited|abl
american business council|abc
american centres|american centre
american civil liberties union|aclu
american israeli public affairs committee|aipac
american muslim council|amc
amnesty international|ai
anjuman tajiran sindh|ats
ansar burney welfare trust international|abwti
ansar burney welfare trust|abwt
anti narcotic force|anf
anti terrorism court|atc
app corporation|app corporation
arab league|al
argentina cricket association|aca
army medical college|amc
army public schools|army public school
army strategic forces command|asfc
army welfare trust|awt
asean regional forum|arf
asian cooperation|asian co
asian cricket foundation|acf
asian development bank|adb
asian development fund|adf

asian pacific economic conference|apec
asian parliamentarians|asian parliamentarian
asian squash federation|asf
askari information systems|ais
ataullah mengal|ata mengal
auqaf department|auqaf dept
australia cricket board|acb
australian broadcasting corporation|abc
australian cricket board|acb
austrialia cricket board|acb
awami action committee|aac
awami league|al
awami national conference|anc
awami national party|anp
ayub medical college|amc
babri masjid action committee|bmac
babri masjid movement coordination committee|bmmcc
balochistan high court|bhc
balochistan national movement|bnm
balochistan national party|bnp
balochistan olympic association|boa
baluchistan assembly|baluch assembly
baluchistan national party|bnp
bank of china|boc
bank of khyber|bok
bank of punjab|bop
banking department|banking dept
banking services corporation|bsc
bar associations|bar association
barani area development project|badp
berkeley nucleonic corporation|bnc
bhabha atomic research center|barc

bhabha atomic research centre|barc
bharat sanchar nigam limited|bsnl
bharatiya jana sangh|bjs
bharatiya janata party|bjp
bharatiya janta party|bjp
bhartia janta party|bjp
bhartyia janata party|bjp
blue berrets|blue berret
bolan medical college|bmc
boxing champion|boxing champ
boy scouts associations|boy scouts association
british airways|ba
british broadcast company|bbc
british broadcasting corporation|bbc
british council|bc
bush administration|bush administration
business software alliance|bsa
business support centre|bsc
cabinet committee on privatisation|ccop
cable news network|cnn
canadian international development agency|cida
capital development authority|cda
caribbean media corporation|cmc
carrier telephone industries|cti
central executive committee|cec
central intelligence agency|cia
central intelligence department|cid
central police office|cpo
central reserve police force|crpf
central superior services|css
chief election commission|cec
china gold association|cga

china national petroleum corporation|cnpc

china radio international|cri

china software industry association|csia

cholistan development authority|cda

citizen community board|ccb

citizen peace committee|cpc

citizens peace committee|cpc

civil aviation authority|caa

clipsal pakistan limited|cpl

coalition information center|cic

commonwealth development corporation|cdc

commonwealth ministerail action group|cmag

commonwealth science council|csc

community welfare attache|cwa

comprehensive economic partnership|cep

comsats institute of information technology|comsat institute of information technology

comstech information technology centers|comstech information technology center

congressional budget office|cbo

counsel generals|counsel general

cricket world cup|cwc

criminal appeals|criminal appeal

ctv national news|cnn

cultural counsellor|cultural co

defence consultative group|dcg

defence division|d division

defence intelligence agency|dia

defence production|dp

defence research development organisation|drdo

defense consultative group|dcg

defense housing authority|dha

democratic freedom party|dfp

democratic liberation party|dlp
democratic national committee|dnc
democratic political movement|dpm
department for international development|dfid
department of defence|dod
department of defense|dod
department of justice|dept of justice
deutche welle|dw
deutsche welle|dw
dhaka international trade fair|ditf
diphosphate amonium phosphate|dap
dir area support project|dasp
directorates of intelligence|directorate of intelligence
district cricket association|dca
district management group|dmg
district public safety commission|dpsc
economic advisory board|eab
economic affairs division|ead
economic cooperation organisation|eco
economic cooperation organization|eco
economic coordination committee|ecc
economic coordination organization|eco
economic management group|emg
election commissioners|ec
election commissioners|election commission
election commissioners|election commission|ec
election commission|ec
election tribunals|et
electronic data systems|eds
electronic government directorate|egd
emerging sciences|emerging science
emirates cricket board|ecb

engineering development board|edb
england cricket board|ecb
environment protection agency|epa
environment protection department|epd
environmental protection agency|epa
environmental protection|environment protection
environmental tribunals|environment tribunals
eu council|ec
euorpean union|eu
european commission|ec
european commission|eu commission
european parliamentarians|european parliament
european parliament|ep
european parliament|eu parliament
european unions|european union|eu
european union|eu
export development fund|edf
export facilitation center|efc
farhad company|fc
farm machinery institute|fmi
fauji fertilizer company limited|ffcl
fauji fertilizer company|ffc
federal and provincial governments|federal and provincial government
federal council|fc
federal court|fc
federal government services hospital|fgsh
federal intelligence agency|fia
federal investigation authority|fia
federal investigative agency|fia
federal public service commission|fpsc
federal public services commission|fpsc
federal security bureau|fsb

federal services|federal service
federal shariat court|fsc
federal tax ombudsman|fto
federal trade commission|ftc
finance departments|finance department
finance department|finance dept|fd
financial action task force|fatf
financial advisors|fa
first information reports|fir
first information report|fir
food department|food dept
foreign currency exchange services|fces
foreign direct investment|fdi
foreign ministry|fm
foreign office|fo
foreign services|foreign service
foreign trade ands export corporation|foreign trade and export corporation
forest department|forest dept
gawadar development authority|gda
general electric|ge
general motors|gm
general sales tax|gst
ghazi barotha hydro power project|gbhpp
ghazi barotha taraqiati idara|gbti
global broadcast network|gbn
global change impact studies centre|gcisc
global information technology|git
global water partnership|gwp
government medical college|gmc
government of india|goi
governments of india|government of india
gulf cooperation council countries|gccc

gulf cooperation council|gcc
habib bank limited|hbl
habib exchange company|hec
hajj policy|haj policy
hajj visa|haj visa
hamriyah free zone|hfz
haq parast group|hpg
hattar industrial estate|hie
hazara qaumi mohaz|hqm
head constable|hc
high commissioners|high commission
high commissioners|high commission|hc
high commission|hc
high court bar association|hcba
high court|hc
hindustan times|ht
house building finance corporation|hbfc
house of commons|house of common
house of lords|house of lord
human development forum|hdf
human development foundation|hdf
human development fund|hdf
human resource development network|hrdn
human rights bureau|hrb
human rights forum|hrf
human rights foundation|hrf
human rights watch|hrw
hunza development forum|hdf
hurriyat conference|hc
hurriyet conference|hc
illinois national guardsman|illinois national guard
independent press association|ipa

indus river system authority|irsa
indus water commissioners|indus water commission
indus water commission|iwc
ineternational cricket council|icc
information communications technology|ict
information technology commerce network|itcn
institute of business management|iobm
intelligence bureau|ib
intelligence burueau|ib
inter service intelligence|isi
inter services intelligence)|isi
inter services intelligence|isi
international air transport association|iata
international air travel association|iata
international airlines transport association|iata
international atomic energy agency|iaea
international boxing federation|ibf
international civil aviation organization|icao
international cricket conference|icc
international cricket council|icc
international criminal court|icc
international development association|ida
international finance corporation|ifc
international human rights commission|ihrc
international islamic university|iiu
international labor organization|ilo
international labour organisation|ilo
international labour organistaion|ilo
international labour organization|ilo
international maritime bureau|imb
international maritime organization|imo
international monetary fund|imf

international monitoring fund|imf
international olympic committee|ioc
international olympic council|ioc
international olympics committee|ioc
international political forum|ipf
international relations committee|irc
international rescue committee|irc
international security forces|isf
international watch organization|iwo
international women moot|iwm
ipu) council|ipu council
islami jamhoori ittehad|iji
islamic society of statistical sciences|isoss
islamic students league|isl
israel defense forces|idf
israeli defence forces|idf
jamhoori watan party|jwp
jammu and kashmir national panthers party|jaknpp
jamshoro power company|jpc
japan cultural associations|japan cultural association
japan emergency ngos|jen
jawaharlal nehru sports trust|jnst
jeay sindh qaumi mahaz|jsqm
jeay sindh qaumi movement|jsqm
jeay sindh quami mahaz|jsqm
jet propulsion laboratory|jpl
jinnah postgraduate medical center|jpmc
jiye sindh qaumi mahaz|jsqm
justice department|justice dept
kahota research laboratories|krl
kahuta research laboratories|krl
karakuram international university|kiu

kashmiri american council|kac
kashmiri scandinavian council|ksc
khan research laboratories|krl
khushal pakistan program|kpp
khushali bank|kb
khushhali bank|kb
khyber medical college|kmc
kinetic warhead|kw
king edward medical college|kemc
kings party|king party
korean broadcasting system|kbs
kuwait petroleum company|kpc
kyrgyzstan caa|kyrgyz caa
land acquisition committee|lac
latif akbar|la
layton rehmatulla benevolent trust|lrbt
legislative assembly|la
lever brothers pakistan limited|lbpl
lexus gss|lexus gs
liberal democratic party|ldp
liberal forum pakistan|lfp
library of congress|loc
lions clubs|lions club
liquefied petroleum gas|lpg
local area network|lan
local governments|local government
local government|lg
loya jirga commission|ljc
management development services|mds
marine rescue coordination center|mrcc
maritime security agency|msa
marylebone cricket club|mcc

mas freedom foundation|mas freedom foundation
mehsuds of badar|mehsud of badar
merv dillon|m dillon
metallurgical construction company|mcc
microsoft corporation|microsoft corp
millat party|mp
milli yakjehti council|myc
ministry of communications|ministry of communication
ministry of defence|mod
ministry of health|moh
ministry of industries and productions|ministry of industries and production
ministry of information|ministry of i
ministry of women development|mowd
minority advisory council punjab|macp
mirpur development authority|mda
mna hostels|mna hostel
mohajir qaumi movement|mqm
mutaheda qaumi movement|mqm
mutahhida qaumi movement|mqm
mutahidda qaumi movement|mqm
muttaheda qaumi movement|mqm
muttahida jehad council|mjc
muttahida qaumi movement|mqm
muttahida quami movement|mqm
muttahida quomi movement|mqm
muttehida quami movement|mqm
national accountability bureau|nab
national agricultural research centre|narc
national agricultural research council|narc
national agricultural research systems|nars
national alliance;|national alliance
national alliancehave|national alliance

national awami party pakistan|napp
national basketball association|nba
national book foundation|nbf
national broadcasting corporation|nbc
national command authority|nca
national conference|nc
national construction limited|ncl
national cricket academy|nca
national defence college|ndc
national democratic alliance|nda
national development finance corporation|ndfc
national drainage project|ndp
national economic commission|nec
national economic council|nec
national education foundation|nef
national education policy|nep
national electoral commission|nec
national electric power regulatory authority|nepra
national environmental action plan|neap
national fertiliser corporation|nfc
national fertilisers corporation|nfc
national fertilizer corporation|nfc
national finance commission|nfc
national financial commission|nfc
national fisheries development board|nfdb
national food authority|nfa
national front|nf
national highway authority|nha
national human rights commission|nhrc
national indicative programe|nip
national institute health|nih
national investment trust|nit

national logistic cell|nlc
national olympic committee|noc
national people congress|npc
national police academy|npa
national police bureau|national police burea
national power construction company|npcc
national power construction corporation|npcc
national reconstruction bureau|nrb
national reconstuction bureau|nrb
national security advisory board|nsab
national security agency|nsa
national security council|nsc
national security force|nsf
national technology incubators|nti
national umpires council|national umpire council
national water policy|nwp
national workers party|nwp
natural resource management|nrm
naval armament depot|nad
neonatal tetanus|nt
nepali congress|nc
network leasing corporation limited|nlcl
new california media|ncm
new south wales|nsw
new york police department|nypd
new york stock exchange|nyse
new york times|nyt
new zealand cricket|nzc
nh house|nh
non government organizations|ngo
non governmental organisations|ngo
non governmental organization|ngo

north waziristan agency|nwa
northern alliance|north alliance
northern area legislative council|nalc
nwfp agriculture department|nwfp agri dept
office management group|omg
oic conference|oic co
oil companies advisory committee|ocac
oil company advisory committee|ocac
oilfields limited|oilfield limited
oilseed development board|oil development board
overseas employment corporation|oec
overseas pakistan foundation|opf
overseas pakistanis advisory council|opac
overseas pakistanis foundation|opf
overseas private investment corporation|opic
overseas private investment corp|opic
pac advisory council|pac
pacific news service|pns
pakhtoonkhawa milli awami party|pmap
pakistan agricultural research council|parc
pakistan agriculture research council|parc
pakistan air forces|paf
pakistan air fore|paf
pakistan american business association|paba
pakistan american congress|pac
pakistan american council|pac
pakistan american democratic forum|padf
pakistan armed forces|paf
pakistan armed forces|pakistan armed force
pakistan armed services board|pasb
pakistan atomic energy commission|paec
pakistan bar council|pbc

pakistan boxing federation|pbf
pakistan bridge federation|pbf
pakistan broadcasting corporation|pbc
pakistan community services|pcs
pakistan computer bureau|pcb
pakistan cricket board|pcb
pakistan education foundation|pef
pakistan educational research network|pern
pakistan electrical fans manufacturers association|pefma
pakistan electronic media regulatory authority|pemra
pakistan engineering council|pec
pakistan environment protection foundation|pepf
pakistan girls guide association|pgga
pakistan golf federation|pgf
pakistan green task force|pgtf
pakistan high commission|phc
pakistan housing authority|pha
pakistan human development fund|phdf
pakistan human rights commission|phrc
pakistan industrial development corporation|pidc
pakistan international airlines corporation|piac
pakistan international airlines|pia
pakistan international human rights organization|pihro
pakistan islamic medical association|pima
pakistan karate federation|pkf
pakistan law commission|plc
pakistan medical association|pma
pakistan meteorological department|pmd
pakistan microfinance network|pmn
pakistan muslim league|pml
pakistan national accreditation council|pnac
pakistan national aids program|pnap

pakistan news serviceeditorial|pakistan news service
pakistan news service|pns
pakistan olympic association|poa
pakistan oppressed national alliance movement|ponam
pakistan ordnance factory|pof
pakistan people party|ppp
pakistan petroleum dealers association|ppda
pakistan petroleum limited|ppl
pakistan pharmaceutical manufacturers association|ppma
pakistan red crescent society|prcs
pakistan science foundation|psf
pakistan security printing corporation|pspc
pakistan software export board|pseb
pakistan sports board|psb
pakistan squash federation|psf
pakistan standard institution|psi
pakistan state oil|pso
pakistan steel|ps
pakistan students associations|pakistan students association
pakistan tehreek insaf|pti
pakistan tehrik insaf|pti
pakistan tele communication limited|ptcl
pakistan telecom authority|pta
pakistan telecommunications authority|pta
pakistan telecommunications company limited|ptcl
pakistan telecommunications corporation limited|ptcl
pakistan tobacco board|ptb
pakistan tourism development cooperation|ptdc
pakistan tourism development corporation|ptdc
pakistan trade centre|ptc
pakistan water partnership|pwp
pakistan welfare society|pws

pakistan workers federation|pwf
palestinian broadcasting corp|pbc
palestinian liberation organisation|plo
palestinian national authority|pna
peshawar high court|phc
pharmaceutical industry|pharma industry
pilotless target aircraft|pta
political action committee|pac
ppp punjab|pp
privatisation commission|pc
professional squash association|psa
progressive state oil|pso
progressive women association|pwa
provincial finance commission|pfc
provincial public safety commission|ppsc
public adhoc committee|pac
public affairs committee|pac
public information department|pid
public offering of ssgcl|public offering of ssgc
pukhtoonkhwa milli awami party|pmap
punjab health foundation|phf
punjab olympic association|poa
punjab provincial cooperative bank limited|ppcbl
punjab small industries corporation|psic
punjab squash association|psa
punjab tourism development corporation|ptdc
punjab university|pu
qaumi jamhoori party|qjp
qaumi jamhrooi party|qjp
qaumi tajir ittehad|qti
radio organizations|radio organization
rashtriya swayam sevak|rss

rashtriya swayamsevak sangh|rss
rashtriya swayamsewak sangh|rss
rawalpindi cricket association|rca
rawalpindi medical college|rmc
red crescent authority|rca
red crescent society|rcs
red cross|rc
regional accountability bureau|rab
regional development finance corporation|rdfc
regional organizations|regional organization
rehman medical institute|rmi
religious organizations|religious organization
republican guards|republican guard
revolutionary united front|ruf
rohri limited|rohri limited
royal air force|raf
royal canadian mounted police|rcmp
royal united services institute|rusi
rural health center|rhc
rural social development program|rsdp
saarc human resource development centre|shrdc
saarc information centre|sic
saarc summits|saarc summit
saf media cell|saf media cel
sarhad development authority|sda
saudi electric company|saudi electric com
saudi press agency|spa
saudi royal air forces|sraf
save) committee|save committee
science and technology;|science and technology
sciences and technology|science and technology
sdpi) study group|sdpi study group

security councila|security council
security forces (bsf)|security force (bsf)
shalimar television network|stn
shangahi cooperation orgnaizaton|sco
shanghai cooperation organization|sco
shaukat khanum memorial cancer hospital|skmch
shaukat khanum memorial trust|skmt
shiromani gurdwara parbandhak committee|sgpc
sichuan petroleum administration|spa
sindh adabi board|sab
sindh agriculture university teachers association|sauta
sindh amateur cycling association|saca
sindh assembly|sa
sindh communist party|scp
sindh democratic alliance|sda
sindh engineering limited|sel
sindh environmental protection agency|sindh environment protection agency|sepa
sindh high court|shc
sindh information technology board|sitb
sindh medical college|smc
sindh national front|snf
sindh services hospital|sindh service hospital|ssh
sindh taraqqi passand party|stpp
sindh university|su
singapore electronic system|ses
singapore international airlines|sia
small business finance corporation|sbfc
small industrial development board|sidb
social action program|sap
south asian federation|saf
south asian free media association|safma
south asian olympic council|saoc

south asian sports federation|sasf
south waziristan agency|swa
special air service|sas
special boat squadron|sbs
special communication organisation|sco
special communication organization|sco
special communications organization|special communication organization|sco
special economic zone|sez
special operation group|sog
special operations group|sog
special task forces|stf
special task force|stf
sports board punjab|sbp
sri gurdwara prabandhak board|sgpb
state department|state dept
state engineering corporation|sec
state human rights commission|shrc
strategic forces command|sfc
sui southern gas company limited|ssgcl
sui southern gas company|ssgc
supreme court bar association|scba
taj company|taj co
taliban|taliban
taraqee foundation|tf
technology development fund|tdf
tehsil nazim|tn
telugu desam party|tdp
thai parliamentarian|thai parliament
the citizen foundation|tcf
total medial limited|total media limited
trade development agency|tda
trade union congress|tuc

transportation division|transport division
turkish air forces|turkish air force
turkish naval|turk naval
turkish presidential|turkish president
union council|uc
united airlines|ua
united bank limited|ubl
united christian front|ucf
united cricket board|ucb
united front|uf
united jehad council|ujc
united national party|unp
united nations|un
united nations development program|undp
united nations development project|undp
united nations disaster management team|undmt
united nations industrial development organisation|unido
united nations industrial development organization|unido
united nations information center|unic
united nations international children educational fund|unicef
united nations international children emergency fund|unicef
united nations organization|uno
united nations security council|unsc
united states administration|usa
united states air force|usaf
us agriculture department|us agri dept
us congress|us cong
us governments|us government
us state department|us state dept
utility stores corporation|usc
voice of america|voa
wall street journal|wsj

washington post|wp
water and power development authority|wapda
water and sanitation agency|wasa
water resources research institute|wrri
west indies cricket board|wicb
wetlands project|wetland project
women cricket association|wca
women development foundation|wdf
women health project|whp
women in technology|wit
workers welfare fund|wwf
world bank|wb
world boxing council|wbc
world boxing organisation|wbo
world craft council|wcc
world cricket action committee|wcac
world cup|wc
world economic forum|wef
world economics forum|world economic forum|wef
world food program|wfp
world gold council|wgc
world health assembly|wha
world squash federation|wsf
world tourism organisation|wto
world tourism organization|wto
world trade organisation|wto
world trade organization|wto
world trade organizaton|wto
world wildlife fund|wwf
xiv asiad|xi asiad
yum restaurants international|yri
zarai taraqiati bank limited|ztbl

zarai taraqqiati bank limited|ztbl

ziauddin medical university|zmu

zone vi|zone v

C. LOCATIONS

abaro,sindh

abasna,india

abbattobad,pakistan

abbotabad,bannu

abbotabad,pakistan

abbottabad,pakistan

abottabad,pakistan

african,afghanistan

agam,afghanistan

agra,india

ahmedabad,india

akron,ohio

alabama,usa

alameda,california

alexandria,egypt

alexandria,va

alexandria,virginia

algiers,algeria

almatay,kazakhstan

almaty,kazakhstan

ames,iowa

amman,jordan

amritsar,india

amstelveen,netherlands

amsterdam,holland

amsterdam,netherlands

anaconda,alamo

andijan,uzbekistan
ankara,turkey
antwerp,belgium
arachi,pakistan
arequipa,peru
argentineans,india
arin,bandipora
arkansas,usa
arlington,virginia
asean,laos
aseer,tabuk
asgabat,turkmenistan
asghabat,turkmenistan
ashdod,israel
ashghabad,turkmenistan
assam,iaf
assemblies,pakistan
astore,pakistan
athens,greece
atlanta,georgia
attock,pakistan
aurora,colorado
austin,texas
avenue,brooklyn
ayodhya,india
ayubia,pakistan
badin,pakistan
bagram,afghanistan
bahawalnagar,pakistan
bahawalpur,pakistan
bahawalpur,rs
baisa,iraq

bakersfield,ca
baku,azerbaijan
balakot,pakistan
bali,indonesia
balochistan,pakistan
bam,iran
bamako,mali
bamyan,afghanistan
bandipora,baramulla
bandipora,rajouri
bandypoora,budhwar
bangalore,india
bangkok,thailand
bangok,thailand
bannu,dikhan
baqoubah,iraq
barcelona,spain
barmer,bikaner
basel,switzerland
basha,skardu
batagram,pakistan
beaumont,texas
beijing,china
beirut,lebanon
berkeley,california
bermingham,uk
bhakkar,pakistan
bhalwal,sargodha
bhawalpur,pakistan
bhopal,india
bhuj,india
bhurban,murree

bhurban,pakistan
bhuttan,india
bicester,oxfordshire
biermingham,uk
bingol,turkey
birmingham,uk
bise,peshawar
blain,williamson
boao,china
bogota,colombia
boise,idaho
bolan,sibi
bombay,india
bonavish,balkh
bonn,germany
bonn,western
boston,ma
boston,massachusetts
boumerdes,algeria
brdelas,gilgit
bridgetown,barbados
bridgeview,il
brighton,england
brisbane,australia
brisbane,queensland
bsa,ifpi
bsp,ncp
budapest,hungary
budhal,rajouri
bulawayo,zimbabwe
burewala,pakistan
busan,korea

cairo,egypt
california,usa
cambridge,massachusetts
cambridge,uk
canada,north america
cancum,mexico
cancun,mexico
canterbury,uk
canton,ohio
cardiff,england
carrara,italy
casablanca,morocco
cern,geneva
cerritos,ca
chagai,pakistan
chakala,pakistan
chaklala,pakistan
chakwal,pakistan
chaman,pakistan
chamman,baluchistan
chandigarh,india
chandpur,bangladesh
chantilly,france
charsadda,bannu
charsadda,pakistan
charsadda,sanam
chashma,mianwali
chattragul,kangan
chennai,india
cherbourg,france
chicago,usa
chichawatni,pakistan

china,asia
chinar,pakistan
chishtian,pakistan
chitagong,bangladesh
chithisinghpora,kashmir
chitral,pakistan
cologne,germany
columbia,maryland
columbia,missouri
columbus,georgia
columbus,ohio
comparatively,pakistan
compton,ca
comstech,idb
copenhagen,denmark
cortina,italy
crawford,texas
cudahy,ca
dade,broward
dadu,pakistan
dakar,senegal
dallas,texas
damascus,syria
dammam,riyadh
dar,reserve
dashkin,pakistan
daska,pakistan
daudkhel,mianwali
dc,usa
debkafile,india
deceitfully,india
dehli,india

dehrawad,afghanistan
delhi,india
delhi,sagroor
deoband,india
detroit,mi
detroit,michigan
dha,rawalpindi
dhaka,bangladesh
dhamtore,abbottabad
dhol,baja
diplo,pakistan
diplomatically,india
doda,rajouri
doha,qatar
dover,delaware
dubai,uae
dunyapur,pakistan
dushanbe,tajikistan
edmonton,canada
encino,california
eriteria,ghana
eugene,oregon
evian,france
excellencies,pakistan
fairfax,virginia
faisalabad,pakistan
faislalabad,pakistan
falfurrias,texas
fallujah,iraq
faryab,kunduz
fata,islamabad
fata,pakistan

fata,paksat
fata,peshawar
fcm,nawabshah
fia,karachi
ficci,fpcci
flemington,nj
florida,us
florida,usa
fns,germany
frankfurt,germany
fremont,california
french,russia
fukuoka,japan
gabba,brisbane
galyat,mansehra
gandawah,baluchistan
ganderbal,achabal
gardez,afghanistan
gawadar,meerani
gawadar,pakistan
gazaryar,kupwara
geneva,switzerland
genoa,italy
georgetown,guyana
ghangche,italy
ghangche,pakistan
ghanghce,pakistan
ghazi,tarbela
ghent,belgium
ghotki,jacobabad
ghotki,sukkur
ghottki,jacobabad

gifu,japan
gilgit,pakistan
glasgow,scotland
gniezno,poland
godhra,india
gorakhpur,india
gorny,russia
gpo,islamabad
green,pakistan
guadalajara,mexico
guangdong,china
guangnan,china
guantanamo,cuba
gujranwala,faisalabad
gujranwala,gujrat
gujranwala,pakistan
gujrat,pakistan
gultari,northern
guwahati,india
gwadar,pakistan
gwaliar,india
gwangju,incheon
gwangju,korea
hafizabad,pakistan
hainan,china
hajj,jeddah
halabja,iraq
hamburg,germany
hangu,sharki
hansera,india
haripur,pakistan
harlan,iowa

haroonabad,pakistan
harpsund,sweden
hartford,connecticut
hassanabdal,pakistan
hattian,attock
hayatabad,peshawar
hellar,kokernage
helmand,afghanistan
helmand,uruzgan
herat,bamiyan
hilton,colombo
hiroshima,japan
hollywood,florida
hongkong,thailand
hospital,chitral
houston,texas
hyderabad,district
hyderabad,india
hyderabad,pakistan
ibrat,hyderabad
igp,sindh
illinois,usa
ilo,undp
indus,gilgit
interprise,sukkur
inzaman,pakistan
ioskom,turkey
ipoh,malaysia
ipps,wapda
isalamabad,pakistan
isamabad,pakistan
isi,pakistan

iskandariyah,iraq
islamabad,pakistan
islamabad,pakitan
islamabad,patten
islambad,pakistan
islmabad,pakistan
istanbul,turkey
iucn,pakistan
izmir,turkey
jaccobabad,pakistan
jacobabad,pakistan
jaffarabad,pakistan
jaglot,pakistan
jakarat,indonesia
jakarta,indonesia
jalalabad,afghanistan
jalozai,pakistan
jauharabad,pakistan
jauhrabad,pakistan
jawans,naval
jeddah,saudi
jehlum,pakistan
jenin,ramallah
jhang,multan
jhang,pakistan
jhudo,pakistan
jiuquan,gansu
jkpp,ppp
joharabad,pakistan
jomtien,thailand
jordan,israel
jowzjan,sar

juharabad,pakistan
jullundur,india
kabul,afghanistan
kaduna,nigeria
kafaragua,switzerland
kakul,pakistan
kanadahar,afghanistan
kananaskis,alberta
kandahar,afghanistan
karachi,pakistan
karahci,quetta
karak,kohat
kasur,pakistan
kathmandu,nepal
katich,bracken
kauai,hawaii
kazapunga,wana
khairpur,pakistan
khairpur,sindh
khairpur,sukkur
khanewal,pakistan
khanghar,pakistan
khanpur,nasirabad
khanpur,pakistan
kharan,kalat
kharan,pakistan
kharian,pakistan
khartoum,sudan
kheri,aarpinchla
kiev,ukraine
kingsmead,durban
kingston,jamaica

kita,japan
kiwis,pakistan
koenigswinter,germany
kohala,pakistan
kohat,pakistan
kohlu,sui
kolgam,islamabad
kong,dubai
korollelvu,fiji
kotli,pakistan
krakow,poland
kulgam,islamabad
kumble,india
kundi,kupwara
kunduz,bamiyan
kyoto,japan
laeken,belgium
lagos,nigeria
lahore,pakistan
larkana,dadu
larkana,jacobabad
larkana,pakistan
larkana,shikarpur
larkana,sibi
larkana,sindh
larkana,sukkur
larnaca,cyprus
lausanne,switzerland
layyah,pakistan
lcci,malaysian
leeds,england
leicester,uk

lillee,england
liloan,philippines
lima,peru
linkoping,sweden
lisbon,portugal
logar,khost
logar,uruzgan
lomita,ca
london,england
london,uk
lords,england
lucknow,india
ludhiana,india
luna,chitral
madison,wisconsin
madras,india
madras,rss
madrid,spain
makung,taiwan
male,maldives
malikwal,pakistan
manama,bahrain
manama,us
manchester,england
manchester,uk
mandra,pakistan
mangla,pakistan
mango,citrus
manila,philippines
manshera,pakistan
mardan,kohat
mardan,pakistan

mashad,iran
mashud,iran
masjid,india
masnaa,lebanon
mazar,jalalabad
melawa,afghanistan
melbourne,australia
memphis,tennessee
merida,mexico
miami,florida
mianwali,pakistan
middlesbrough,england
midland,texas
mills,pakistan
minfal,cotton
mingora,pakistan
mirpur,muzaffarabad
mirpur,pakistan
mirpurkhas,pakistan
mithi,pakistan
mobile,alabama
modena,italy
monterrey,mexico
montgomery,alabama
montreal,canada
moscow,russia
mosul,iraq
mou,pakistan
multan,pakistan
mumbai,india
munich,germany
murree,pakistan

muslimabad,kohat
muzafargarh,pakistan
muzaffarabad,pakistan
muzaffargarh,pakistan
naantali,finland
nacogdoches,texas
nagpur,india
nahrin,afghanistan
nairobi,kenya
najaf,iraq
najaf,karbala
nakial,pakistan
nankana,pakistan
narawara,eidgah
narowal,pakistan
nashville,tn
nasiriyah,iraq
nastogaz,ukraine
nawabshah,pakistan
nazimabad,karachi
nevada,usa
newark,california
newlands,capetown
newton,massachusetts
newyork,tokyo
newyork,usa
ngos,afghanistan
nha,gwadar
nimmud,leh
nlc,rawalpindi
noonday,texas
norfolk,virginia

northridge,ca
northrop,ca
nottingham,england
nottingham,uk
nowshera,pakistan
nyon,switzerland
oakland,california
ocona,peru
odense,denmark
oita,japan
orlando,florida
osaka,japan
oslo,norway
otocari,bosnia
ottawa,canada
paf,pakistan
paigham,sahiwal
pakistan,asia
pakpattan,pakistan
paktia,khost
palestine,texas
panaji,india
paramaribo,suriname
paris,france
paro,bhutan
pashtoons,afghanistan
pashtuns,afghanistan
pennsylvania,usa
perth,australia
peshawar,pakistan
peshwar,pakistan
peterborough,england

phagvel,india
philadelphia,pennsylvania
philadelphia,usa
phoenix,arizona
piedar,islamabad
pims,polyclinic
piplan,mianwali
pishin,quetta
pitham,karachi
pittsburgh,pa
pittsburgh,pennsylvania
pittsfield,massachusetts
pomona,ca
poonch,rajouri
portland,oregon
pota,india
prgf,pakistan
proteas,pakistan
pseb,sindh
punjab,sindh
punjab,india
punjab,province
pwd,quetta
qadirpur,kadanwari
qala,afghanistan
qalat,afghanistan
qandahar,afghanistan
qatari,azerbaijan
quetta,pakistan
rabat,morocco
rafiganj,india
ragistan,india

raipur,india
raiwind,pakistan
rajasthan,india
rajasthan,tihar
ramstein,germany
rawalpindi,pakistan
rawlakot,pakistan
reading,pennsylvania
reston,virginia
richardson,texas
riga,latvia
risalpur,pakistan
riverside,ca
rochester,ny
rome,italy
roses,pakistan
rosoboronexport,india
rotherham,england
rotterdam,holland
rss,india
ruwani,afghanistan
sacramento,california
saddar,karachi
sadiqabad,pakistan
safta,pakistan
sahahar,basra
sahiwal,pakistan
sailkot,sindh
saindak,pakistan
sakardu,hunza
sanghar,pakistan
sanglahill,pakistan

saraj,afghanistan
sargodha,pakistan
sargodha,rs
sarhad,abbottabad
sarobi,afghanistan
sarriab,quetta
satkhira,jessore
scottsdale,arizona
sehwan,pakistan
seibersdorf,austria
shakargarh,pakistan
shanghai,china
shankiari,pakistan
shanksville,us
sharjah,uae
shatoi,chechnya
sheffield,england
sheikhupura,pakistan
shimla,tashkent
shizuoka,japan
shkargarh,pakistan
sialkot,pakistan
sialkot,rs
sibi,pakistan
sigonella,italy
siliguri,india
skardo,pakistan
skardu,pakistan
skims,soura
slamabad,pakistan
smeda,pakistan
snamprogetti,italy

soham,england
springfield,virginia
srilanka,algeria
srinagar,india
sringar,kashmir
stephenville,texas
stn,shalimar
stockholm,sweden
strangely,india
strasbourg,france
stuttgart,germany
suitland,maryland
sujaat,sindh
sukkur,pakistan
sunnyvale,california
sust,pakistan
suwon,korea
sydney,australia
tacoma,washington
talagang,pakistan
taleban,afghanistan
tampa,florida
tangier,morocco
tarnab,peshawar
tarragona,spain
tashkent,uzbekistan
taxila,pakistan
tehran,iran
tehsildar,abbottabad
texas,usa
thimpu,bhutan
thirdly,india

ti,india
tikrit,iraq
tokyo,japan
toronto,canada
township,domail
trieste,italy
tripoli,libya
tufnell,croft
turin,italy
tustin,ca
uav,india
ubl,pakistan
udine,italy
ueberlingen,germany
ultan,pakistan
unfpa,italy
unido,iib
united states,north america
urumqi,china
vacaville,california
valencia,spain
valladolid,spain
vancouver,canada
venice,italy
verrettes,haiti
vettori,warne
vienna,austria
vienna,va
vientiane,laos
vilseck,germany
waddington,england
wagah,pakistan

wah,pakistan
wana,pakistan
warcha,kalabagh
washington dc,united states
wasington,united states
waziristan,pakistan
westmont,illinois
weybridge,uk
whistle,pakistan
windies,india
xian,china
xinjiang,china
yeman,zimbabwe
yogyakarta,indonesia
yokohama,japan
zairat,kalat
zanakote,hmt
zangalopora,islamabad
zhejiang,china
zhob,pakistan
ziarat,pakistan
zte,pakistan

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX B

A. SUPPORT CLASSES JAVADOC

1. Class FileHandler

java.lang.Object

└ **thesis.FileHandler**

All Implemented Interfaces:

qdxml.DocHandler

---

public class **FileHandler**

extends java.lang.Object

implements qdxml.DocHandler

Author:

Matthew W. Esparza

Constructor Summary
`FileHandler()`

Method Summary	
`void`	`close()`

void	**endDocument**()
void	**endElement**(java.lang.String elem)
java.lang.String	**getDirectoryName**()
java.lang.String	**getFileName**()
int	**getLineCount**()
java.lang.String	**getPath**()
void	**init**(java.lang.String fileName)
java.lang.String[]	**listDirectoryContents**()
void	**openDirectory**()
java.lang.String	**openFileChannel**(java.lang.String specFile)
void	**setLineCount**(int num)
void	**startDocument**()
void	**startElement**(java.lang.String elem,

	java.util.Hashtable h)
void	text(java.lang.String text)

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

FileHandler

public **FileHandler**()

Method Detail

init

public void **init**(java.lang.String fileName)

---

close

public void **close**()

throws java.io.IOException

Throws:

`java.io.IOException`

---

openFileChannel

public java.lang.String **openFileChannel**(java.lang.String specFile)

---

openDirectory

public void **openDirectory**()

---

listDirectoryContents

public java.lang.String[] **listDirectoryContents**()

---

getFileName

public java.lang.String **getFileName**()

---

getPath

public java.lang.String **getPath**()

---

getDirectoryName

public java.lang.String **getDirectoryName**()

---

setLineCount

public void **setLineCount**(int num)

---

getLineCount

public int **getLineCount**()

---

startDocument

public void **startDocument**()

throws java.lang.Exception

Specified by:

`startDocument` in interface `qdxml.DocHandler`

Throws:

`java.lang.Exception`

---

endDocument

public void **endDocument**()

Specified by:

`endDocument` in interface `qdxml.DocHandler`

---

startElement

public void **startElement**(java.lang.String elem,

java.util.Hashtable h)

Specified by:

`startElement` in interface `qdxml.DocHandler`

---

endElement

public void **endElement**(java.lang.String elem)

Specified by:

`endElement` in interface `qdxml.DocHandler`

---

text

public void **text**(java.lang.String text)

Specified by:

`text` in interface `qdxml.DocHandler`

2. Class Person

java.lang.Object

└ **thesis.Person**

---

public class **Person**

extends java.lang.Object

Author:

Matthew W. Esparza

Constructor Summary

`Person()`

Method Summary

`void`	`addActions(java.lang.String str)`
`void`	`addAssoc(java.lang.String str)`
`void`	`addCity(java.lang.String str)`

void	**addContent**(java.lang.String str)
void	**addCountry**(java.lang.String str)
void	**addEnemies**(java.lang.String str)
void	**addFamily**(java.lang.String str)
void	**addFriends**(java.lang.String str)
void	**addLocation**(java.lang.String str)
void	**addName**(java.lang.String str)
void	**addOrganization**(java.lang.String str)
void	**addQuotes**(java.lang.String str)
void	**addReference**(java.lang.String str)
void	**addTimelineData**(java.util.Calendar date)
void	**addTitle**(java.lang.String str)

void	addTombstoneData(java.lang.String str)
java.util.LinkedList	getActions()
java.util.LinkedList	getAssocs()
java.util.LinkedList	getCity()
java.util.LinkedList	getContent()
java.util.LinkedList	getCountry()
java.util.LinkedList	getEnemies()
java.util.LinkedList	getFamily()
java.util.LinkedList	getFriends()
java.util.LinkedList	getLocation()
java.util.regex.Pattern	getNamePattern()
java.util.LinkedList	getNames()
java.util.LinkedList	getOrganizations()

java.util.LinkedList	**getQuotes**()
java.util.LinkedList	**getReferences**()
java.util.LinkedList	**getTimelineData**()
java.util.LinkedList	**getTitles**()
java.util.LinkedList	**getTombstoneData**()
boolean	**isVoid**()
java.lang.String[]	**removeRedundancy**(java.lang.String[] array)
void	**setNamePattern**(java.lang.String str)

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

Person

public **Person**()

Method Detail

isVoid

public boolean **isVoid**()

---

addName

public void **addName**(java.lang.String str)

---

getNames

public java.util.LinkedList **getNames**()

---

addTitle

public void **addTitle**(java.lang.String str)

---

getTitles

public java.util.LinkedList **getTitles**()

---

addTombstoneData

public void **addTombstoneData**(java.lang.String str)

---

getTombstoneData

public java.util.LinkedList **getTombstoneData**()

---

addTimelineData

public void **addTimelineData**(java.util.Calendar date)

---

getTimelineData

public java.util.LinkedList **getTimelineData**()

---

addFamily

public void **addFamily**(java.lang.String str)

---

getFamily

public java.util.LinkedList **getFamily**()

---

addFriends

public void **addFriends**(java.lang.String str)

---

getFriends

public java.util.LinkedList **getFriends**()

---

addAssoc

public void **addAssoc**(java.lang.String str)

---

getAssocs

public java.util.LinkedList **getAssocs**()

---

addEnemies

public void **addEnemies**(java.lang.String str)

---

getEnemies

public java.util.LinkedList **getEnemies**()

---

addQuotes

public void **addQuotes**(java.lang.String str)

---

getQuotes

public java.util.LinkedList **getQuotes**()

---

addActions

public void **addActions**(java.lang.String str)

---

getActions

public java.util.LinkedList **getActions**()

---

addCountry

public void **addCountry**(java.lang.String str)

---

getCountry

public java.util.LinkedList **getCountry**()

---

addCity

public void **addCity**(java.lang.String str)

---

getCity

public java.util.LinkedList **getCity**()

---

addLocation

public void **addLocation**(java.lang.String str)

---

getLocation

public java.util.LinkedList **getLocation**()

---

addOrganization

public void **addOrganization**(java.lang.String str)

---

getOrganizations

public java.util.LinkedList **getOrganizations**()

---

addReference

public void **addReference**(java.lang.String str)

---

getReferences

public java.util.LinkedList **getReferences**()

---

addContent

public void **addContent**(java.lang.String str)

---

getContent

public java.util.LinkedList **getContent**()

---

setNamePattern

public void **setNamePattern**(java.lang.String str)

---

getNamePattern

public java.util.regex.Pattern **getNamePattern**()

---

removeRedundancy

public java.lang.String[] **removeRedundancy**(java.lang.String[] array)

3. Class TupleArray

java.lang.Object

└ **thesis.TupleArray**

---

public class **TupleArray**

extends java.lang.Object

Author:

Matthew W. Esparza

Constructor Summary

TupleArray(int num)

Method Summary

void	**addString**(java.lang.String s)
java.lang.String	**argMax**()
java.lang.String	**argMin**()
double	**calcPer**(double num)
int	**getCount**(int index)
int	**getIndex**(java.lang.String s)
double	**getPercent**(int index)
double	**getSize**()
java.lang.String	**getString**(int index)

void	**incrementCount**(java.lang.String s)
double	**logProb**(double percentAsDec)
void	**removeString**(java.lang.String s)
void	**setCount**(int num, int index)
void	**setPercent**(double num, int index)
void	**setString**(java.lang.String s, int index)
boolean	**valuePresent**(java.lang.String s)

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

TupleArray

public **TupleArray**(int num)

Method Detail

getSize

public double **getSize**()

---

addString

public void **addString**(java.lang.String s)

---

removeString

public void **removeString**(java.lang.String s)

---

valuePresent

public boolean **valuePresent**(java.lang.String s)

---

incrementCount

public void **incrementCount**(java.lang.String s)

---

setString

public void **setString**(java.lang.String s,

int index)

---

getString

public java.lang.String **getString**(int index)

---

getIndex

public int **getIndex**(java.lang.String s)

---

setCount

public void **setCount**(int num,

int index)

---

getCount

public int **getCount**(int index)

---

argMax

public java.lang.String **argMax**()

---

argMin

public java.lang.String **argMin**()

---

setPercent

public void **setPercent**(double num,

int index)

---

getPercent

public double **getPercent**(int index)

---

calcPer

public double **calcPer**(double num)

---

logProb

public double **logProb**(double percentAsDec)

B. INFORMATION EXTRACTION JAVADOC

1. Class Ordering

java.lang.Object

└ Ordering

---

public class Ordering

extends java.lang.Object

Author:

Matthew W. Esparza

Constructor Summary
Ordering()

Method Summary	
static void	init(java.lang.String[] args)

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

Ordering

public Ordering()

Method Detail

init

public static void init(java.lang.String[] args)

Parameters:

args –

2. Class ReferenceCounter

java.lang.Object

└ **ReferenceCounter**

All Implemented Interfaces:

qdxml.DocHandler

---

public class **ReferenceCounter**

extends java.lang.Object

implements qdxml.DocHandler

Author:

Matthew W. Esparza

Constructor Summary

ReferenceCounter()

Method Summary

void	**endDocument**()
void	**endElement**(java.lang.String elem)
static int	**getCounter**()
static java.lang.String	**getVal**()
static void	**init**(java.lang.String[] args)
static void	**processNumbers**(java.lang.String str)
static java.lang.String[]	**retDocList**()
static void	**setCounter**(int num)
static void	**setVal**(java.lang.String str)

void	**startDocument**()
void	**startElement**(java.lang.String elem, java.util.Hashtable h)
void	**text**(java.lang.String text)

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

ReferenceCounter

public **ReferenceCounter**()

Method Detail

startDocument

public void **startDocument**()

throws java.lang.Exception

Specified by:

`startDocument` in interface `qdxml.DocHandler`

Parameters:

`null` -

Throws:

`java.lang.Exception`

---

endDocument

public void **endDocument**()

Specified by:

`endDocument` in interface `qdxml.DocHandler`

Parameters:

`null` -

---

startElement

public void **startElement**(java.lang.String elem,

java.util.Hashtable h)

Specified by:

`startElement` in interface `qdxml.DocHandler`

Parameters:

`String` - elem, Hashtable h

---

endElement

public void **endElement**(java.lang.String elem)

Specified by:

`endElement` in interface `qdxml.DocHandler`

Parameters:

`String` - elem

---

text

public void **text**(java.lang.String text)

Specified by:

`text` in interface `qdxml.DocHandler`

Parameters:

`String` - text

---

setVal

public static void **setVal**(java.lang.String str)

Parameters:

`String` - str

---

getVal

public static java.lang.String **getVal**()

Parameters:

`null` -

Returns:

void

---

processNumbers

public static void **processNumbers**(java.lang.String str)

Parameters:

`String` - str

---

retDocList

public static java.lang.String[] **retDocList**()

Parameters:

`null` -

Returns:

String[]

---

setCounter

public static void **setCounter**(int num)

Parameters:

`int` - num

---

getCounter

public static int **getCounter**()

Parameters:

`null` -

Returns:

int

---

init

public static void **init**(java.lang.String[] args)

Parameters:

`String` - args[]

3. Class Searcher

java.lang.Object

└ **Searcher**

---

public class **Searcher**

extends java.lang.Object

Author:

Matthew W. Esparza

Constructor Summary

`Searcher()`

Method Summary

`static void`	`init(java.lang.String[] args)`
`static void`	`marchThrough(java.lang.String[] files)`

Methods inherited from class java.lang.Object

`equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructor Detail

Searcher

public **Searcher**()

Method Detail

marchThrough

public static void **marchThrough**(java.lang.String[] files)

Parameters:

`files` -

---

init

public static void **init**(java.lang.String[] args)

throws java.lang.Exception

Parameters:

`args` -

Throws:

`java.lang.Exception`

C. KEYWORD EXTRACTION JAVADOC

1. Class KeywordExtractor

java.lang.Object

└ **SenFil4.KeywordExtractor**

---

public class **KeywordExtractor**

extends java.lang.Object

Constructor Summary

`KeywordExtractor()`

Method Summary

`static void`	`collectFamily(java.lang.String sentence, java.lang.String firstName, java.lang.String lastName, thesis.Person p)`
`static void`	`collectFriends(java.lang.String sentence, thesis.Person p)`
`static void`	`collectName(java.lang.String sentence, java.lang.String firstName, java.lang.String lastName, thesis.Person p)`

static void	**collectOrganizations**(java.lang.String sentence, thesis.Organization currOrg, thesis.Person p)
static boolean	**collectQuote**(java.lang.String sentence, thesis.Person p)
static void	**collectTitle**(java.lang.String sentence, java.lang.String firstName, java.lang.String lastName, java.lang.String currTitle, thesis.Person p)
static void	**decideLocation**(thesis.Person p)
static void	**decideName**(thesis.Person p)
static void	**decideOrganization**(thesis.Person p)
static void	**decideTitle**(thesis.Person p)
static void	**initLocationPatterns**(java.lang.String[] locs)
static void	**initOrganPatterns**(java.lang.String[] orgs)
static java.lang.String[]	**IsolateEntities**(java.lang.String fileContent)
static void	**main**(java.lang.String[] args)

Methods inherited from class java.lang.Object
`equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructor Detail

KeywordExtractor

public **KeywordExtractor**()

Method Detail

initOrganPatterns

public static void **initOrganPatterns**(java.lang.String[] orgs)

---

initLocationPatterns

public static void **initLocationPatterns**(java.lang.String[] locs)

IsolateEntities

public static java.lang.String[] **IsolateEntities**(java.lang.String fileContent)

Parameters:

`Takes` - no parameters.

Returns:

Returns an array of entity names and the sentences for each entity.

---

collectName

public static void **collectName**(java.lang.String sentence,

java.lang.String firstName,

java.lang.String lastName,

thesis.Person p)

Parameters:

`String[]` - array -- Sentences from file. Person p -- Person object for current person.

---

decideName

public static void **decideName**(thesis.Person p)

---

collectTitle

public static void **collectTitle**(java.lang.String sentence,

java.lang.String firstName,

java.lang.String lastName,

java.lang.String currTitle,

thesis.Person p)

---

decideTitle

public static void **decideTitle**(thesis.Person p)

---

collectQuote

public static boolean **collectQuote**(java.lang.String sentence,

thesis.Person p)

---

collectOrganizations

public static void **collectOrganizations**(java.lang.String sentence,

thesis.Organization currOrg,

thesis.Person p)

---

decideOrganization

public static void **decideOrganization**(thesis.Person p)

---

collectFamily

public static void **collectFamily**(java.lang.String sentence,

java.lang.String firstName,

java.lang.String lastName,

thesis.Person p)

---

collectFriends

public static void **collectFriends**(java.lang.String sentence,

thesis.Person p)

---

decideLocation

public static void **decideLocation**(thesis.Person p)

---

main

public static void **main**(java.lang.String[] args)

Parameters:

`args` -

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX C

Source code:

Ordering.java

```
import thesis.FileHandler;
import thesis.Person;
import java.util.Iterator;

/**
 * @author Matthew W. Esparza
 * @date   2007
 * @class  Ordering.java
 * @description This class is responsible for ordering the entity references.
 */

public class Ordering {

	private static FileHandler fh = new FileHandler();

	/**
	 * @name	init
	 * @param	args
	 * @return	void
	 * @input	The input opened by the openFileChannel function is a
	 *		file containing the entity references.
	 * @output	The output is a file containing the entity references in
	 *		order.
	 *
	 */

	public static void init(String[] args) {

		String name = null;

		fh.init("MORETHAN100_080207_SORTED.txt");

		String content = fh.openFileChannel("MORETHAN100_080207.txt");
		String[] entityBlocks	= content.split("p_");
		Person[] perArray	= new Person[entityBlocks.length];

		// Step 1: Get all of the entities in Person objects
		for(int p = 0; p < entityBlocks.length; p++){

			perArray[p] = new Person();

			String[] lines = entityBlocks[p].split("\n");

			for(int t = 0; t < lines.length; t++){

				if(lines[t].contains("is referenced")){

					name = lines[t].substring(0,
							lines[t].indexOf("_p"));
					perArray[p].addName(name);
```

```
                }
                else if(lines[t].contains("START REFERENCES")){

                        while(lines[t].contains("END REFERENCES") ==
                                                                false){

                            String[] numArray = lines[t+1].split(" ");
                            for(int n = 0; n < numArray.length; n++){

                               if(numArray[n].matches("[0-9]+")){

                                perArray[p].addReference(numArray[n]);
                               }
                            }

                            t++;
                        }
                }
        }
}

String[] per1Refs, per2Refs;

// SORTING
for(int s = 0; s < perArray.length; s++){

        per1Refs = new String[perArray[s].getReferences().size()];
        Iterator p1 = perArray[s].getReferences().iterator();
        int idx = 0;

        while(p1.hasNext()){

                per1Refs[idx] = (String)p1.next();
                idx++;
        }

        int loValue = -1;
        if(per1Refs.length > 0 && per1Refs[0] != null){

                loValue = Integer.parseInt(per1Refs[0]);

        }

        for(int r = s+1; r < perArray.length; r++){

         per2Refs = new String[perArray[r].getReferences().size()];
         Iterator p2 = perArray[r].getReferences().iterator();
         int idx2 = 0;

                while(p2.hasNext()){

                        per2Refs[idx2] = (String)p2.next();
                        idx2++;
                }

                if(per2Refs.length > 0 && per2Refs[0] != null){

                        int val2 = Integer.parseInt(per2Refs[0]);

                        if(val2 < loValue){

                            // Swap
```

```
                                        Person temp = perArray[s];
                                        perArray[s] = perArray[r];
                                        perArray[r] = temp;

                                        loValue = val2;
                                }
                        }
                }
        }

        // PRINTING
        Iterator l = null, m = null;
        for(int i = 0; i < perArray.length; i++){

                if(perArray[i] != null && perArray[i].getNames().isEmpty()
                                                                        == false)
                {
                        l = perArray[i].getNames().iterator();

                        while(l.hasNext()){

                          String str = (String)l.next();
                          System.out.println("p_" + str + " is referenced " +
                                        perArray[i].getReferences().size() +
                                           " times.");
                          System.err.println(str);
                        }

                        m = perArray[i].getReferences().iterator();
                        int count = 0;
                        System.out.println("START REFERENCES");

                        while(m.hasNext()){

                                String ref = (String)m.next();
                                if(count % 10 == 0 && count != 0)
                                        System.out.println();
                                else
                                        System.out.print(ref + " ");
                        }

                        System.out.println();
                        System.out.println("END REFERENCES");
                        System.out.println();
                }
        }

        try{
                fh.close();
        }
        catch (Exception e){

                e.printStackTrace();
        }
    }

}
```

ReferenceCounter.java

```
import java.io.*;
import java.util.Enumeration;
import java.util.Hashtable;
import thesis.FileHandler;
import qdxml.*;

/**
 *
 * @author          Matthew W. Esparza
 * @date            2007
 * @name            ReferenceCounter
 * @description     Class implements the DocHandler interface from the Quick
 *                  and Dirty XML Parser.  Bascially, given a minimum and
 *                  maximum number as thresholds, the class will search through
 *                  UniqueEntities.xml and retrieve the entities that have the
 *                  desired number of references.
 */

public class ReferenceCounter implements DocHandler {

       private static final int                MIN_LIMIT = 99;
       private static final int                MAX_LIMIT = 10000000;
       private static int                      entityCount = 0, count;
       private static String                   key, val, docList;
       private static String[]                 initial;
       private static ReferenceCounter         rfc = new ReferenceCounter();
       private static FileHandler              fh  = new FileHandler();

       /* -- Starts implentation of DocHandler -- */

       /**
        *  @name    startDocument
        *  @param  null
        *  @return void
        */
       public void startDocument() throws Exception {

       }

       /**
        *  @name    endDocument
        *  @param  null
        *  @return void
        */
       public void endDocument() {

       }

       /**
        *  @name            startElement
        *  @param    String elem, Hashtable h
        *  @return   void
        *  @description Parses the XML and stores the elements in a hashtable.
        */
       public void startElement(String elem, Hashtable h) {

              Enumeration e = h.keys();
```

```
        while (e.hasMoreElements()) {

                key = (String) e.nextElement();
                val = (String) h.get(key);

                key.trim();

                setVal(val);
        }
}

/**
 *  @name    endElement
 *  @param  String elem
 *  @return void
 */
public void endElement(String elem) {


         /* if(elem.equalsIgnoreCase("PERSON")){

                System.out.println(" end elem: " + elem);

          }*/

}

/**
 *  @name    text
 *  @param  String text
 *  @return void
 */
public void text(String text) {

        docList = text;
        processNumbers(docList);
}

/* -- Ends implementation of DocHandler -- */

/**
 *  @name    setVal
 *  @param  String str
 *  @return void
 */
public static void setVal(String str){

        val = str;
}

/**
 *  @name    getVal
 *  @param  null
 *  @return void
 */
public static String getVal(){

        return val;
}
```

```
/**
 *  @name           processNumbers
 *  @param          String str
 *  @return         void
 *  @description    Loops through UniqueEntities.xml and prints out
 *                  reference indexes of entities
 *                  that meet the limit criteria.
 */
public static void processNumbers(String str) {

      initial = str.split(" ");

      int counter = 0;

      for (int i = 0; i < initial.length; i++) {

            counter++;
      }

      setCounter(counter);

      if(counter > MIN_LIMIT && counter <= MAX_LIMIT){

            if(initial[0].trim().length() > 0){

                  System.out.println(getVal() + " is referenced " +
                                          getCounter() + " times.\n");
                  System.out.println("START REFERENCES");

                  for (int j = 0; j < initial.length; j++) {

                        if(j % 10 == 0 && j != 0)
                              System.out.println();
                        else
                              System.out.print(initial[j] + " ");
                  }

                  System.out.println();
                  System.out.println("END REFERENCES");
                  System.out.println();
                  entityCount++;
            }
      }
}

/**
 *  @name    retDocList
 *  @param  null
 *  @return String[]
 */
public static String[] retDocList(){

      return initial;
}

/**
 *  @name    setCounter
 *  @param  int num
 *  @return void
 */
public static void setCounter(int num) {
```

```
		count = num;
	}

	/**
	 *  @name    getCounter
	 *  @param  null
	 *  @return int
	 */
	public static int getCounter() {

		return count;
	}

	/**
	 *  @name           init
	 *  @param          String args[]
	 *  @return         void
	 *  @description    Parses XML document through the use of quick and
	 *                  dirty parser, which in turn calls the reference
	 *                  related functions above.
	 */
	public static void init(String args[]) {

		fh.init("MoreThan100.txt");

		// Open UniqueEntities.xml
		fh.openFileChannel("UniqueEntities.xml");

		try {
			FileReader fr = new FileReader("UniqueEntities.xml");
			QDParser.parse(rfc, fr);
		}

		catch (Exception e) {

			e.printStackTrace();
		}

		System.out.println(entityCount + " entities");

		try{

			fh.close();
		}
		catch (IOException ioe){

			ioe.printStackTrace();
		}
	}
}
```

Searcher.java

```
import java.util.regex.*;
import java.util.Iterator;
import thesis.FileHandler;
import thesis.Person;

/**
 * @author          Matthew W. Esparza
 * @date            2007
 * @name            Searcher
 * @description     This class is responsible for finding the reference numbers
 *                  output by the ReferenceCounter class.  It searches through
 *                  multiple files in the corpus directory.
 */

public class Searcher {

	private static String[]    entityBlocks;
	private static FileHandler fh = new FileHandler();
	private static Person[]    perArr;

	/**
	 * @name            marchThrough
	 * @param           files
	 * @return          void
	 * @description     Scans through XML corpus of tagged news articles and
	 *                  looks for <PERSON> tags.  If the ID in the tag
	 *                  matches the one we are looking for, then we collect
	 *                  the sentence it was found in and add it to the
	 *                  current person's "content block."
	 */
	public static void marchThrough(String[] files){

		Pattern      perId   = Pattern.compile("PERSON ID=\"[0-9]+\"");
		Matcher      idMatch = null;
		String       path    = null;
		String       content = null;

		for(int e = 0; e < files.length; e++){

			path = fh.getDirectoryName() + "\\" + files[e];
			System.err.println("Now processing: " + path);

			content = fh.openFileChannel(path);
			String[] lines = content.split("\n");
			boolean firstTime = true;

			// Will find instance of "Person ID="XX" in XML files
			for(int j = 0; j < lines.length; j++){

				idMatch = perId.matcher(lines[j]);

				while(idMatch.find()){

					String       mention = idMatch.group();
					int    firstQuote = mention.indexOf("\"");
					int    lastQuote  = mention.lastIndexOf("\"");

					String refNumb = mention.substring(firstQuote
```

```
                                        + 1, lastQuote);
                        Int    menNumb = Integer.parseInt(refNumb);
                        boolean found  = false;

                        for(int l = 0; l < perArr.length; l++){

                                if(perArr[l] != null){

                                        Iterator itr =
                                perArr[l].getReferences().iterator();


                                        while(itr.hasNext()){

                                                int ref =
                                Integer.parseInt((String)itr.next());

                                                if(menNumb == ref){

                                                        if(ref ==
                        Integer.parseInt((String)perArr[l].getReferences().get(0))){

                                                                firstTime =
                                                                        true;
                                                        }
                                                        else{

                                                                firstTime =
                                                                        false;
                                                        }

                                                        found = true;


if(perArr[l].getNames().size() > 0 && firstTime == true){


System.out.println("Name: " + perArr[l].getNames().get(0) + "\n");
                                                                firstTime =
                                                                        false;
                                                        }

                                                        String noTags =
                                                lines[j].replaceAll("<.*?>", "");

                                                perArr[l].addContent(noTags);

                                                        break;
                                                }
                                        }

                                Iterator con = perArr[l].getContent().iterator();

                        String[] conArr = new String[perArr[l].getContent().size()];

                                        int conCount = 0;

                                        while(con.hasNext()){

                                conArr[conCount] = (String)con.next();
```

```
                                        }

                                        if(conArr != null){
                                                String[] newConArr =
                                    perArr[l].removeRedundancy(conArr);

                                                for(int o = 0; o <
                                                newConArr.length; o++){


                                        System.out.println(newConArr[o]);
                                                }

                                        }
                                    }

                                    if(found == true){

                                        break;
                                    }
                                }
                            }
                        }

                        lines = null;
                }
        }

        /**
         * @name            init
         * @param           args
         * @throws          Exception
         * @return    void
         * @description     Loops through the XML corpus and calls marchThrough.
         */
        public static void init(String[] args) throws Exception{

                fh.init("MORETHAN100_080207_SORTED_NOTAGS.txt");

                // Read Input File and split into sentences
                String name     = null;
                String content  = fh.openFileChannel
                                        ("MORETHAN100_080207_SORTED.txt");

                entityBlocks    = content.split("p_");
                perArr          = new Person[entityBlocks.length];

                fh.openDirectory();
                String[] fileListing = fh.listDirectoryContents();

                // Process references
                for(int p = 0; p < entityBlocks.length; p++){

                        perArr[p] = new Person();

                        String[] lines = entityBlocks[p].split("\n");

                        for(int t = 0; t < lines.length; t++){

                                if(lines[t].contains("is referenced")){

                                        name = lines[t].substring(0,
```

```
                                        lines[t].indexOf(" is"));

                                if(name != null && name.length() > 0){
                                        perArr[p].addName(name);
                                }
                        }
                        else if(lines[t].contains("START REFERENCES")){

                                while(lines[t].contains("END REFERENCES") ==
                                                                        false){

                                        String[] numArray = lines[t +
                                                                1].split(" ");

                                        for(int n = 0; n < numArray.length;
                                                                        n++){

                                                if(numArray[n].matches("[0-
                                                                9]+")){

                perArr[p].addReference(numArray[n]);
                                                }
                                        }

                                        t++;
                                }
                        }
                }
        }

        marchThrough(fileListing);

        try{
                fh.close();
        }
        catch (Exception e){

                e.printStackTrace();
        }
    }
}
```

KeywordExtractor.java

```
import java.util.regex.*;
import java.util.Iterator;
import thesis.Person;
import thesis.TupleArray;
import thesis.FileHandler;

/**
 * @class           KeywordExtractor
 * @author          Matthew W. Esparza
 * @description     Contains functions to fill slots of biography.
 *                  Slots include Name, Job Title, Nationality, Tombstone Data,
 *                  Professional Relationships, Familial Relationships,
 *                  Enemies, Friends, Quotes [not counted as truth], Actions
 * @version         4.0
 */

public class KeywordExtractor {

      private static FileHandler  fh          = new FileHandler();
      private static Person[]     people;
      private static Pattern[]    orgPats, contPats, cityPats, titlePats;

      public static Pattern[] initOrganPatterns(String[] orgs){

            int    numOrgs      =      orgs.length;
            orgPats             =      new Pattern[numOrgs];

            for(int z = 0; z < numOrgs; z++){

                  String[]currOrg     = orgs[z].split("\\|");
                  String  orgName     = currOrg[0];
                  orgName             = orgName.replaceAll("\\p{Punct}", "");
                  String  abbrev      = currOrg[1];
                  abbrev              = abbrev.replaceAll("\\p{Punct}", "");
                  orgPats[z]          = Pattern.compile(" (" + orgName + ") |
                                              (" + abbrev + ") ",
                                              Pattern.DOTALL |
                                              Pattern.MULTILINE |
                                              Pattern.CASE_INSENSITIVE);
            }

            return orgPats;
      }

      public static Pattern[] initCountryPatterns(String[] locations){

            int    numLocas     =      locations.length;
            contPats            =      new Pattern[numLocas];

            for(int z = 0; z < numLocas; z++){

                  String[]currLoc     = locations[z].split(",");
                  String  countryName = currLoc[1].trim();
                  contPats[z]         = Pattern.compile(" (" + countryName +
                                              ") ", Pattern.DOTALL |
                                              Pattern.MULTILINE |
                                              Pattern.CASE_INSENSITIVE);
            }
```

```
        return contPats;
}

public static Pattern[] initCityPatterns(String[] locations){

        int           numLocas      =      locations.length;
        cityPats                          =      new Pattern[numLocas];

        for(int z = 0; z < numLocas; z++){

                String[]currLoc           = locations[z].split(",");
                String  cityName    = currLoc[0].trim();
                cityPats[z]         = Pattern.compile(" (" + cityName + ")
                                          ", Pattern.DOTALL |
                                          Pattern.MULTILINE |
                                          Pattern.CASE_INSENSITIVE);
        }

        return contPats;
}

public static Pattern[] initTitlePatterns(String[] titles){

        int    numTitles          =      titles.length;
        titlePats                 =      new Pattern[numTitles];

        for(int z = 0; z < numTitles; z++){

                String currTitle    = titles[z];

                // Some titles like "pace-bowler" have an embedded dash
                // that needs to be turned into a literal "\\-" for the
                // purpose of the regex.
                String reFormTitle  = currTitle.replaceAll("\\-", "\\\\-");
                titlePats[z]        = Pattern.compile(" ?" + reFormTitle +
                                          " ", Pattern.DOTALL |
                                          Pattern.MULTILINE |
                                          Pattern.CASE_INSENSITIVE);
        }

        return titlePats;
}

/**
 * @name IsolateEntities
 * @purpose Reads string (content of file) and extracts contents of next
 *          Entity by reading everything between Name: and the next Name:
 *
 * @param     Takes no parameters.
 * @return    Returns an array of entity names and the sentences for each
 *            entity.
 */
public static String[] IsolateEntities(String fileContent) {

        // Will give you all the content between the colon in the first
        // Name:
        // Until it reaches the next Name:
        // SHOULD correspond to the number of entities collected in file.
        String[] entities = fileContent.split("Name:");

        return entities;
```

```
}

/**
 * @name      collectNames
 * @purpose   Reads strings (sentences from file) and extracts names of
 *            Entity by reading everything between given first name and
 *            last name.
 *
 * @param     String[] array -- Sentences from file.
 *                  Person p       -- Person object for current person.
 * @return    Function does not return anything.  It does, however,
 *            output to file specified in main.
 */
public static void collectName(String sentence, String firstName, String
                                                lastName, Person p){

      String regex   = null;

      if(lastName != null){
            regex = "(" + firstName + ")" + ".*?" + "(" + lastName +
                                                                  ")";
      }
      else
            regex = "(" + firstName + ")";

      // Compile a regular expression to look for name (with anything in
      // between first and last)
      Pattern fullName = Pattern.compile(regex, Pattern.CASE_INSENSITIVE
                                                | Pattern.DOTALL |
                                                Pattern.MULTILINE);
      Matcher match    = fullName.matcher(sentence);

      while(match.find()){

            // Store current string found and remove any excess
            // characters
            String current = match.group();
                     current = current.replaceAll("\\)|\\]|^
                                    ?[^\\p{Alpha}\\p{Space}]*", "");

            if(current.length() > 30){

               String newReg = "[^" + firstName + "]" + firstName +
                                                      ".*?" + lastName;
               Pattern preciseName     = Pattern.compile(newReg,
                                             Pattern.CASE_INSENSITIVE |
                                             Pattern.DOTALL |
                                             Pattern.MULTILINE);
               Matcher preciseMatch = preciseName.matcher(current);

               while(preciseMatch.find()) {

                        String preciseCurrent = preciseMatch.group();
                        preciseCurrent =
                              preciseCurrent.replaceAll("\\)|^
                                    ?[^\\p{Alpha}\\p{Space}]*", "");

                        // Last ditch effort to grab name
                        if(preciseCurrent.length() > 30){

                           String  simpReg = firstName +  lastName;
```

```
                                    Pattern simpleName =
                                                Pattern.compile(simpReg,
                                                      Pattern.CASE_INSENSITIVE |
                                                      Pattern.DOTALL |
                                                      Pattern.MULTILINE);

                                    Matcher simpleMatch =
                                                simpleName.matcher(current);

                                    while(simpleMatch.find()) {

                                                String simpCurrent  =
                                                      simpleMatch.group();
                                                simpCurrent =
                                                      simpCurrent.replaceAll("^
                                                ?[^\\p{Alpha}\\p{Space}]*", "");


                                                if(simpCurrent != null){
                                                      p.addName(simpCurrent);
                                                }
                                          }
                                    }
                                    else {

                                          if(preciseCurrent != null){
                                                p.addName(preciseCurrent);
                                          }
                                    }
                              }
                        }
                        else {
                                    if(current != null){
                                          p.addName(current);
                                    }
                        }
                  }
            }

            public static void decideName(Person p){

                  Iterator            t      = p.getNames().iterator();
                  TupleArray          tArray = new TupleArray(p.getNames().size());

                  while(t.hasNext()){

                        tArray.addString((String)t.next());
                  }

                  System.out.println("<NAME>");
                  System.out.println("<DECISION>" + tArray.argMax()+"</DECISION>");
                  System.out.println("</NAME>");
            }

            public static void collectTitle(String sentence, Pattern currPat, Person
                                                                                    p){

                  // Loop to match all of the patterns.
                  Matcher titleMatcher = currPat.matcher(sentence);

                  while(titleMatcher.find()){
```

```
                // Store current string found and remove any excess
                // characters
                String current = titleMatcher.group();

                if(current != null){
                   current = current.replaceAll("^
                                    ?[^\\p{Alpha}\\p{Space}]*", "");
                   p.addTitle(current);
                }
        }
}

public static void decideTitle(Person p){

        Iterator            t      = p.getTitles().iterator();
        TupleArray          tArray = new TupleArray(p.getTitles().size());
        int                 count  = 0;

        while(t.hasNext()){

                tArray.addString((String)t.next());
                count++;
        }

        System.out.println("<TITLE>");
        System.out.println("<DECISION>" + tArray.argMax()+"</DECISION>");
        System.out.println("</TITLE>");
}

public static boolean collectQuote(String sentence, Person p){

        boolean quoteFound         = false;
        String quotation           = null;
        String author              = null;

        Pattern quotes = Pattern.compile("(\".*\")", Pattern.MULTILINE |
                            Pattern.DOTALL);

        Pattern whoSaid = Pattern.compile("\\w* \\w*?
                            ?(?=said|says|stated)", Pattern.DOTALL |
                            Pattern.MULTILINE | Pattern.CASE_INSENSITIVE);

        Matcher quoteMatch  = quotes.matcher(sentence);
        Matcher whoSaidMatch = whoSaid.matcher(sentence);

        while(whoSaidMatch.find()){

                author = whoSaidMatch.group();
                while(quoteMatch.find()){

                        quotation = quoteMatch.group();

                        if(author != null && author != " " && quotation !=
                                                                    null){

                                String quoteInfo = author + " : " + quotation;
                                quoteInfo = quoteInfo.replaceAll("^
                                            ?[^\\p{Alpha}\\p{Space}]*", "");
                                quoteFound = true;
                        }
                }
        }
```

```
	return quoteFound;
}

public static void collectLifespanData(String sentence, Person p){

	Pattern birthWords  = Pattern.compile("\\w* \\w*?
				?(?=born|birth)", Pattern.MULTILINE |
				Pattern.CASE_INSENSITIVE);

	Pattern deathWords  = Pattern.compile("\\w* \\w*?
				?(?=died|death|passed away)",
				Pattern.MULTILINE |
				Pattern.CASE_INSENSITIVE);

	Pattern years       = Pattern.compile("[0-9]{4}?",
				Pattern.MULTILINE);

	Matcher matchBirth  = birthWords.matcher(sentence);
	Matcher matchDeath  = deathWords.matcher(sentence);
	Matcher matchYears  = years.matcher(sentence);

	String tombInfo     = null;

	while(matchBirth.find()){
		tombInfo = matchBirth.group();
		if(tombInfo != null){
			p.addTombstoneData(tombInfo);
		}
	}
	while(matchDeath.find()){
		tombInfo = matchDeath.group();
		if(tombInfo != null){
			p.addTombstoneData(tombInfo);
		}
	}
	while(matchYears.find()){
		if(matchYears.group() != null){
			p.addTombstoneData(sentence);
		}
	}
}

public static void collectFamily(String sentence, Person p){

	Pattern familyWords = Pattern.compile("(family )|(father
				)|(grandfather )|(grandmother )|" +
				"(in(-| )law)|(mother )|(sister
				)|(brother )", Pattern.CASE_INSENSITIVE
				| Pattern.MULTILINE);

	Matcher matchFam    = familyWords.matcher(sentence);

	while(matchFam.find()){

		p.addFamily(sentence);
	}
}
```

```
public static void collectFriends(String sentence, Person p){

	Pattern meetTerms = Pattern.compile("meeting |meet |summit |met
				|gather " + "|assemble |met with
				|gathering " + "|assembled ",
				Pattern.CASE_INSENSITIVE |
				Pattern.MULTILINE);

	Matcher matchMeet = meetTerms.matcher(sentence);

	while(matchMeet.find()){

		p.addFriends(sentence);
	}
}

private static void collectCountry(String sentence, Pattern currLocPat,
                                                        Person p){

	Matcher locMatcher = currLocPat.matcher(sentence);

	while(locMatcher.find()){

		p.addCountry(locMatcher.group());
	}
}

public static void decideCountry(Person p){

	Iterator     t      = p.getCountry().iterator();
	TupleArray   tArray = new TupleArray(p.getCountry().size());
	int          count  = 0;

	while(t.hasNext()){

		tArray.addString((String)t.next());
		count++;
	}

	System.out.println("<COUNTRY>");
	System.out.println("<DECISION>" + tArray.argMax()+"</DECISION>");
	System.out.println("</COUNTRY>");
}

private static void collectCity(String sentence, Pattern currLocPat,
                                                        Person p){

	Matcher locMatcher = currLocPat.matcher(sentence);

	while(locMatcher.find()){

		p.addCity(locMatcher.group());
	}
}

private static void decideCity(Person p){

	Iterator            t      = p.getCity().iterator();
	TupleArray          tArray = new TupleArray(p.getCity().size());
	int                 count  = 0;

	while(t.hasNext()){
```

```
                tArray.addString((String)t.next());
                count++;
        }

        System.out.println("<CITY>");
        System.out.println("<DECISION>" + tArray.argMax() +"</DECISION>");
        System.out.println("</CITY>");
}

public static void collectOrganizations(String sentence, Pattern
                                                currOrgPat, Person p){

        Matcher orgMatcher = currOrgPat.matcher(sentence);

        while(orgMatcher.find()){
                p.addOrganization(orgMatcher.group());
        }
}

public static void decideOrganization(Person p){

        Iterator     t = p.getOrganizations().iterator();
        TupleArray   tArray = new TupleArray(p.getOrganizations().size());
        int          count  = 0;

        while(t.hasNext()){

                tArray.addString((String)t.next());
                count++;
        }

        System.out.println("<ORGANIZATION>");
        System.out.println("<DECISION>" + tArray.argMax() +"</DECISION>");
        System.out.println("</ORGANIZATION>");
}

/**
 * @name main
 * @param args
 * @return Returns nothing.
 */
public static void main(String[] args) {

        // Initialize Output File
        fh.init("BioData_081307.xml");
        System.out.println("<?xml version=\"1.0\" encoding=\"ISO-8859-
                                                                1\"?>");

        // Read Input File and split into entity chunks
        String content = fh.openFileChannel("MORETHAN100_ORIG.txt");
        String[] entities   = IsolateEntities(content);
        int numEntities     = entities.length;
        people              = new Person[numEntities];

        // Open file that contains titles from corpus
        String titleFile    = fh.openFileChannel("Titles.txt");
        String[] titles     = titleFile.split("\r\n");
        int numTitles       = titles.length;

        // Open file that contains locations from corpus
        String locFile      = fh.openFileChannel("Locations.txt");
```

```
String[] locales    = locFile.split("\r\n");
int numLocales      = locales.length;

// Open file that contains organizations from corpus
String orgFile      = fh.openFileChannel("Organizations.txt");
String[] orgs       = orgFile.split("\r\n");
int numOrgs         = orgs.length;

// Init Location, Title, and Organization patterns
initTitlePatterns(titles);
initCountryPatterns(locales);
initCityPatterns(locales);
initOrganPatterns(orgs);

// Store each line of the entity's content for processing.
System.out.println("<BIOGRAPHIES>");

for(int p = 1; p < numEntities; p++){

       System.out.println("<PERSON ID=\"" + p + "\">");
       System.err.println("Now serving number: " + p);

       people[p]                  = new Person();
       String[] entContent        = entities[p].split("\n");

       // Determine name of current entity.
       // Every name starts with a blank space, so start at 1
       String name = entContent[0].substring(1,
                                  entContent[0].length() - 3);
       String firstName           = null;
       String lastName            = null;
       int    numSentences = 0;

       if(name.contains(" ")){
              firstName = name.substring(0, name.indexOf(" "));
              lastName  = name.substring(name.indexOf(" ") + 1,
                                                    name.length());
              numSentences = entContent.length;
       }
       else{
              firstName = name.substring(0, name.length() - 3);
              numSentences = entContent.length;
              System.err.println(firstName);
       }

       for(int s = 1; s < numSentences; s++){

              String currentSentence = entContent[s];
              if(currentSentence.contains("Now looking at reference
                     number ") || currentSentence.contains
                     ("Information from corpus") ||
                     currentSentence.length() == 0){

                     continue;
              }

              // Collect quotes: if one is found, then the current
              // sentence is a quote and we should skip the rest of
              // the loop.
              if(collectQuote(currentSentence, people[p])){
                     continue;
              }
```

```
        // Collect name
        collectName(currentSentence, firstName, lastName,
                                                people[p]);

        // Collect title
        for(int t = 0; t < numTitles; t++){
                collectTitle(currentSentence, titlePats[t],
                                                people[p]);
        }

        // Collect tombstone information
        collectLifespanData(currentSentence, people[p]);

        // Collect family information
        collectFamily(currentSentence, people[p]);

        // Collect friend information
        collectFriends(currentSentence, people[p]);

        // Collect country information
        for(int l = 0; l < numLocales; l++){
                collectCountry(currentSentence, contPats[l],
                                                people[p]);
        }

        // Collect city information
        for(int m = 0; m < numLocales; m++){
                collectCity(currentSentence, cityPats[m],
                                                people[p]);
        }

        // Collect organization information
        for(int n = 0; n < numOrgs; n++){
                collectOrganizations(currentSentence,
                                orgPats[n], people[p]);
        }
}

decideName(people[p]);
decideTitle(people[p]);
decideCountry(people[p]);
decideCity(people[p]);
decideOrganization(people[p]);

System.out.println("<FAMILY>");
Iterator k = people[p].getFamily().iterator();
String[] famArray = new
                    String[people[p].getFamily().size()];
int c  = 0;
while(k.hasNext()){
        famArray[c] = (String)k.next();
        c++;
}

String[] redundFreeFam =
                people[p].removeRedundancy(famArray);

for(int h = 0; h < redundFreeFam.length; h++){

        System.out.println("<F" + h + ">" + redundFreeFam[h]
                                        + "</F" + h + ">");
```

```
                        }
                        System.err.println("Number of family inclusions: " +
                                                        redundFreeFam.length);
                        System.out.println("</FAMILY>");

                        System.out.println("<ASSOCIATES>");
                        Iterator l = people[p].getFriends().iterator();
                        String[] assocArray = new
                                        String[people[p].getFriends().size()];

                        int    d = 0;
                        while(l.hasNext()){
                              assocArray[d] = (String)l.next();
                              d++;
                        }

                        String[] redundFreeAssoc =
                                        people[p].removeRedundancy(assocArray);

                        for(int h = 0; h < redundFreeAssoc.length; h++){

                              System.out.println("<R" + h + ">" +
                                        redundFreeAssoc[h] + "</R" + h + ">");
                        }
                        System.err.println("Number of associate inclusions: " +
                                                        redundFreeAssoc.length);
                        System.out.println("</ASSOCIATES>");
                        System.out.println("</PERSON>");
                  }
                  System.out.println("</BIOGRAPHIES>");

                  try{
                        fh.close();
                  }
                  catch(Exception e){

                        e.printStackTrace();
                  }
            }
}
```

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California